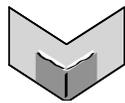


The Journal of Machine Learning Research
Volume 6
Print-Archive Edition

Pages 1099–2204



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 6
Print-Archive Edition

The Journal of Machine Learning Research (JMLR) is an open access journal. All articles published in JMLR are freely available via electronic distribution. This Print-Archive Edition is published annually as a means of archiving the contents of the journal in perpetuity. The contents of this volume are articles published electronically in JMLR in 2005.

JMLR is abstracted in ACM Computing Reviews, INSPEC, and Psychological Abstracts/PsycINFO.

JMLR is a publication of Journal of Machine Learning Research, Inc. For further information regarding JMLR, including open access to articles, visit <http://www.jmlr.org/>.

JMLR Print-Archive Edition is a publication of Microtome Publishing under agreement with Journal of Machine Learning Research, Inc. For further information regarding the Print-Archive Edition, including subscription and distribution information and background on open-access print archiving, visit Microtome Publishing at <http://www.mtome.com/>.

Collection copyright © 2005 The Journal of Machine Learning Research, Inc. and Microtome Publishing. Copyright of individual articles remains with their respective authors.

ISSN 1532-4435 (print)
ISSN 1533-7928 (online)

Editor-in-Chief

Leslie Pack Kaelbling,
*Massachusetts Institute
of Technology, USA*

Managing Editor

Christian R. Shelton, *University of
California at Riverside, USA*

Production Editor

Erik G. Learned-Miller,
*University of Massachusetts,
Amherst, USA*

JMLR Action Editors

Peter Bartlett, *University of California
at Berkeley, USA*

Yoshua Bengio,
Université de Montréal, Canada

Léon Bottou,
NEC Research Institute, USA

Craig Boutilier,
University of Toronto, Canada

Claire Cardie,
Cornell University, USA

David Maxwell Chickering,
Microsoft Research, USA

William W. Cohen,
Carnegie-Mellon University, USA

Nello Cristianini, *UC Davis, USA*

Peter Dayan,
University College, London, UK

Stephanie Forrest,
University of New Mexico, USA

Donald Geman,
Johns Hopkins University, USA

Isabelle Guyon, *ClopiNet, USA*

Ralf Herbrich,
Microsoft Research, Cambridge, UK

Haym Hirsh,
Rutgers University, USA

Aapo Hyvärinen,
University of Helsinki, Finland

Tommi Jaakkola, *Massachusetts Institute
of Technology, USA*

Thorsten Joachims,
Cornell University, USA

Michael Jordan, *University of California
at Berkeley, USA*

John Lafferty,
Carnegie Mellon University, USA

Michael Littman, *Rutgers University, USA*

David Madigan, *Rutgers University, USA*

Sridhar Mahadevan, *University of
Massachusetts, Amherst, USA*

Andrew McCallum, *University of
Massachusetts, Amherst, USA*

Melanie Mitchell,
Oregon Graduate Institute, USA

Fernando Pereira,
University of Pennsylvania, USA

Pietro Perona,
California Institute of Technology, USA

Greg Ridgeway, *RAND, USA*

Dana Ron, *Tel-Aviv University, Israel*

Sam Roweis,
University of Toronto, Canada

Stuart Russell,
University of California at Berkeley, USA

Claude Sammut, *University of
New South Wales, Australia*

Bernhard Schölkopf, *Max-Planck-Institut
für Biologische Kybernetik, Germany*

Dale Schuurmans,
University of Alberta, Canada

John Shawe-Taylor,
Southampton University, UK

Yoram Singer,
Hebrew University, Israel

Manfred Warmuth, *University of
California at Santa Cruz, USA*

Chris Williams,
University of Edinburgh, UK

Stefan Wrobel, *Universität Bonn
and Fraunhofer AIS, Germany*

Bin Yu, *University of California at
Berkeley, USA*

JMLR Editorial Board

Naoki Abe, *IBM TJ Watson
Research Center, USA*

Christopher Atkeson,
Carnegie Mellon University, USA

Andrew G. Barto, *University of
Massachusetts, Amherst, USA*

Jonathan Baxter,
Panscient Pty Ltd, Australia

Richard K. Belew, *University of
California at San Diego, USA*

Tony Bell,
Salk Institute for Biological Studies, USA

Yoshua Bengio,
University of Montreal, Canada

Kristin Bennett,
Rensselaer Polytechnic Institute, USA

Christopher M. Bishop,
Microsoft Research, UK

Lashon Booker,
The Mitre Corporation, USA

Henrik Boström,
Stockholm University/KTH, Sweden

Justin Boyan, *ITA Software, USA*

Ivan Bratko,
Jozef Stefan Institute, Slovenia

Carla Brodley, *Purdue University, USA*

Peter Bühlmann,
ETH Zürich, Switzerland

David Cohn, *Google, Inc., USA*

Walter Daelemans,
University of Antwerp, Belgium

Sanjoy Dasgupta, *University of California
at San Diego, USA*

Luc De Raedt,
University of Freiburg, Germany

Saso Dzeroski,
Jozef Stefan Institute, Slovenia

Usama Fayyad, *DMX Group, USA*

Douglas Fisher,
Vanderbilt University, USA

Peter Flach, *Bristol University, UK*

Nir Friedman, *Hebrew University, Israel*

Dan Geiger, *The Technion, Israel*

Zoubin Ghahramani,
University College London, UK

Sally Goldman,
Washington University, St. Louis, USA

Russ Greiner,
University of Alberta, Canada

David Heckerman,
Microsoft Research, USA

David Helmbold, *University of California
at Santa Cruz, USA*

Geoffrey Hinton,
University of Toronto, Canada

Thomas Hofmann,
Brown University, USA

Larry Hunter,
University of Colorado, USA

Daphne Koller, *Stanford University, USA*

Yi Lin, *University of Wisconsin, USA*

Wei-Yin Loh,
University of Wisconsin, USA

Yishay Mansour,
Tel-Aviv University, Israel

David J. C. MacKay,
Cambridge University, UK

Marina Meila,
University of Washington, USA

Tom Mitchell,
Carnegie Mellon University, USA

Raymond J. Mooney,
University of Texas, Austin, USA

Andrew W. Moore,
Carnegie Mellon University, USA

Klaus-Robert Müller,
University of Potsdam, Germany

Stephen Muggleton,
Imperial College London, UK

Una-May O'Reilly, *Massachusetts
Institute of Technology, USA*

Foster Provost, *New York University, USA*

Lorenza Saitta, *Università del Piemonte
Orientale, Italy*

Lawrence Saul,
University of Pennsylvania, USA

Robert Schapire,
Princeton University, USA

Jonathan Shapiro,
Manchester University, UK

Jude Shavlik,
University of Wisconsin, USA

Satinder Singh,
University of Michigan, USA

Alex Smola, *Australian National
University, Australia*

Padhraic Smyth,
University of California, Irvine, USA

Richard Sutton,
University of Alberta, Canada

Moshe Tennenholtz, *The Technion, Israel*

Sebastian Thrun,
Carnegie Mellon University, USA

Naftali Tishby,
Hebrew University, Israel

David Touretzky,
Carnegie Mellon University, USA

Larry Wasserman,
Carnegie Mellon University, USA

Chris Watkins, *Royal Holloway,
University of London, UK*

- 1 **Asymptotic Model Selection for Naive Bayesian Network**
Dmitry Rusakov, Dan Geiger
- 37 **Dimension Reduction in Text Classification with Support Vector Machines**
Hyunsoo Kim, Peg Howland, Haesun Park
- 55 **Stability of Randomized Learning Algorithms**
Andre Elisseeff, Theodoros Evgeniou, Massimiliano Pontil
- 81 **Learning Hidden Variable Networks: The Information Bottleneck Approach**
Gal Elidan, Nir Friedman
- 129 **Diffusion Kernels on Statistical Manifolds**
John Lafferty, Guy Lebanon
- 165 **Information Bottleneck for Gaussian Variables**
Gal Chechik, Amir Globerson, Naftali Tishby, Yair Weiss
- 189 **Multiclass Boosting for Weak Classifiers**
Günther Eibl, Karl-Peter Pfeiffer
- 211 **A Classification Framework for Anomaly Detection**
Ingo Steinwart, Don Hush, Clint Scovel
- 233 **Denoising Source Separation**
Jaakko Särelä, Harri Valpola
- 273 **Tutorial on Practical Prediction Theory for Classification**
John Langford
- 307 **Generalization Bounds and Complexities Based on Sparsity and Clustering for Convex Combinations of Functions from Random Classes**
Savina Andonova Jaeger
- 341 **A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs**
S. Sathya Keerthi, Dennis DeCoste
- 363 **Core Vector Machines: Fast SVM Training on Very Large Data Sets**
Ivor W. Tsang, James T. Kwok, Pak-Ming Cheung

- 393 **Generalization Bounds for the Area Under the ROC Curve**
*Shivani Agarwal, Thore Graepel, Ralf Herbrich,
Sariel Har-Peled, Dan Roth*
- 427 **Learning with Decision Lists of Data-Dependent Features**
Mario Marchand, Marina Sokolova
- 453 **Estimating Functions for Blind Separation When Sources
Have Variance Dependencies**
Motoaki Kawanabe, Klaus-Robert Müller
- 483 **Characterization of a Family of Algorithms for Generalized
Discriminant Analysis on Undersampled Problems**
Jieping Ye
- 503 **Tree-Based Batch Mode Reinforcement Learning**
Damien Ernst, Pierre Geurts, Louis Wehenkel
- 557 **Learning Module Networks**
Eran Segal, Dana Pe'er, Aviv Regev, Daphne Koller, Nir Friedman
- 589 **Active Learning to Recognize Multiple Types of Plankton**
*Tong Luo, Kurt Kramer, Dmitry B. Goldgof, Lawrence O. Hall,
Scott Samson, Andrew Remsen, Thomas Hopkins*
- 615 **Learning Multiple Tasks with Kernel Methods**
Theodoros Evgeniou, Charles A. Micchelli, Massimiliano Pontil
- 639 **Adaptive Online Prediction by Following the Perturbed Leader**
Marcus Hutter, Jan Poland
- 661 **Variational Message Passing**
John Winn, Christopher M. Bishop
- 695 **Estimation of Non-Normalized Statistical Models
by Score Matching**
Aapo Hyvärinen
- 711 **Smooth ϵ -Insensitive Regression by Loss Symmetrization**
Ofer Dekel, Shai Shalev-Shwartz, Yoram Singer
- 743 **Quasi-Geodesic Neural Learning Algorithms Over the
Orthogonal Group: A Tutorial**
Simone Fiori
- 783 **Machine Learning Methods for Predicting Failures in Hard
Drives: A Multiple-Instance Application**
Joseph F. Murray, Gordon F. Hughes, Kenneth Kreutz-Delgado

- 817 **Multiclass Classification with Multi-Prototype Support Vector Machines**
Fabio Aioli, Alessandro Sperduti
- 851 **Prioritization Methods for Accelerating MDP Solvers**
David Wingate, Kevin D. Seppi
- 883 **Learning from Examples as an Inverse Problem**
Ernesto De Vito, Lorenzo Rosasco, Andrea Caponnetto, Umberto De Giovannini, Francesca Odone
- 905 **Loopy Belief Propagation: Convergence and Effects of Message Errors**
Alexander T. Ihler, John W. Fisher III, Alan S. Willsky
- 937 **Learning a Mahalanobis Metric from Equivalence Constraints**
Aharon Bar-Hillel, Tomer Hertz, Noam Shental, Daphna Weinshall
- 967 **Algorithmic Stability and Meta-Learning**
Andreas Maurer
- 995 **Matrix Exponentiated Gradient Updates for On-line Learning and Bregman Projection**
Koji Tsuda, Gunnar Rätsch, Manfred K. Warmuth
- 1019 **Gaussian Processes for Ordinal Regression**
Wei Chu, Zoubin Ghahramani
- 1043 **Learning the Kernel with Hyperkernels (Kernel Machines Section)**
Cheng Soon Ong, Alexander J. Smola, Robert C. Williamson
- 1073 **A Generalization Error for Q-Learning**
Susan A. Murphy
- 1099 **Learning the Kernel Function via Regularization**
Charles A. Micchelli, Massimiliano Pontil
- 1127 **Analysis of Variance of Cross-Validation Estimators of the Generalization Error**
Marianthi Markatou, Hong Tian, Shameek Biswas, George Hripcsak
- 1169 **Semigroup Kernels on Measures**
Marco Cuturi, Kenji Fukumizu, Jean-Philippe Vert
- 1199 **Separating a Real-Life Nonlinear Image Mixture**
Luis B. Almeida

- 1231 **Concentration Bounds for Unigram Language Models**
Evgeny Drukh, Yishay Mansour
- 1265 **An MDP-Based Recommender System**
Guy Shani, David Heckerman, Ronen I. Brafman
- 1297 **Universal Algorithms for Learning Theory**
Part I : Piecewise Constant Functions
*Peter Binev, Albert Cohen, Wolfgang Dahmen,
Ronald DeVore, Vladimir Temlyakov*
- 1323 **Efficient Computation of Gapped Substring
Kernels on Large Alphabets**
Juho Rousu, John Shawe-Taylor
- 1345 **Clustering on the Unit Hypersphere using
von Mises-Fisher Distributions**
*Arindam Banerjee, Inderjit S. Dhillon,
Joydeep Ghosh, Suvrit Sra*
- 1383 **Inner Product Spaces for Bayesian Networks**
*Atsuyoshi Nakamura, Michael Schmitt,
Niels Schmitt, Hans Ulrich Simon*
- 1405 **Maximum Margin Algorithms with Boolean Kernels**
Roni Khardon, Rocco A. Servedio
- 1431 **A Bayes Optimal Approach for Partitioning the Values
of Categorical Attributes**
Marc Boullé
- 1453 **Large Margin Methods for Structured and Interdependent
Output Variables**
*Ioannis Tschantaridis, Thorsten Joachims,
Thomas Hofmann, Yasemin Altun*
- 1485 **Frames, Reproducing Kernels, Regularization and Learning**
Alain Rakotomamonjy, Stéphane Canu
- 1517 **Local Propagation in Conditional Gaussian Bayesian Networks**
Robert G. Cowell
- 1551 **A Bayesian Model for Supervised Clustering with the Dirichlet
Process Prior**
Hal Daumé III, Daniel Marcu
- 1579 **Fast Kernel Classifiers with Online and Active Learning**
Antoine Bordes, Seyda Ertekin, Jason Weston, Léon Bottou

- 1621 Managing Diversity in Regression Ensembles**
Gavin Brown, Jeremy L. Wyatt, Peter Tino
- 1651 Active Coevolutionary Learning of Deterministic Finite Automata**
Josh Bongard, Hod Lipson
- 1679 Assessing Approximate Inference for Binary Gaussian Process Classification**
Malte Kuss, Carl Edward Rasmussen
- 1705 Clustering with Bregman Divergences**
Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, Joydeep Ghosh
- 1751 Combining Information Extraction Systems Using Voting and Stacked Generalization**
Georgios Sigletos, Georgios Paliouras, Constantine D. Spyropoulos, Michalis Hatzopoulos
- 1783 Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models**
Neil Lawrence
- 1817 A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data**
Rie Kubota Ando, Tong Zhang
- 1855 Feature Selection for Unsupervised and Supervised Inference: The Emergence of Sparsity in a Weight-Based Approach**
Lior Wolf, Amnon Shashua
- 1889 Working Set Selection Using Second Order Information for Training Support Vector Machines**
Rong-En Fan, Pai-Hsuen Chen, Chih-Jen Lin
- 1919 New Horn Revision Algorithms**
Judy Goldsmith, Robert H. Sloan
- 1939 A Unifying View of Sparse Approximate Gaussian Process Regression**
Joaquin Quiñonero-Candela, Carl Edward Rasmussen
- 1961 What's Strange About Recent Events (WSARE): An Algorithm for the Early Detection of Disease Outbreaks**
Weng-Keen Wong, Andrew Moore, Gregory Cooper, Michael Wagner

- 1999 **Change Point Problems in Linear Dynamical Systems**
Onno Zoeter, Tom Heskes
- 2027 **Asymptotics in Empirical Risk Minimization**
Leila Mohammadi, Sara van de Geer
- 2049 **Convergence Theorems for Generalized Alternating Minimization Procedures**
Asela Gunawardana, William Byrne
- 2075 **Kernel Methods for Measuring Independence**
Arthur Gretton, Ralf Herbrich, Alexander Smola, Olivier Bousquet, Bernhard Schölkopf
- 2131 **Efficient Margin Maximizing with Boosting**
Gunnar Rätsch, Manfred K. Warmuth
- 2153 **On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning**
Petros Drineas, Michael W. Mahoney
- 2177 **Expectation Consistent Approximate Inference**
Manfred Opper, Ole Winther

Learning the Kernel Function via Regularization

Charles A. Micchelli

CAM@MATH.ALBANY.EDU

*Department of Mathematics and Statistics
State University of New York
The University at Albany
1400 Washington Avenue
Albany, NY, 12222, USA*

Massimiliano Pontil

M.PONTIL@CS.UCL.AC.UK

*Department of Computer Science
University College London
Gower Street, London WC1E, UK*

Editor: Peter Bartlett

Abstract

We study the problem of finding an optimal kernel from a prescribed convex set of kernels \mathcal{K} for learning a real-valued function by regularization. We establish for a wide variety of regularization functionals that this leads to a convex optimization problem and, for square loss regularization, we characterize the solution of this problem. We show that, although \mathcal{K} may be an uncountable set, the optimal kernel is always obtained as a convex combination of at most $m + 2$ basic kernels, where m is the number of data examples. In particular, our results apply to learning the optimal radial kernel or the optimal dot product kernel.

1. Introduction

A widely used approach to estimate a function from empirical data consists in minimizing a regularization functional in a Hilbert space \mathcal{H} of real-valued functions $f : \mathcal{X} \rightarrow \mathbb{R}$, where \mathcal{X} is a set. Specifically, regularization estimates f as a *minimizer* of the functional

$$Q(I_{\mathbf{x}}(f)) + \mu\Omega(f)$$

where μ is a positive parameter, $I_{\mathbf{x}}(f) = (f(x_j) : j \in \mathbb{N}_m)$ is the *vector* of values of f on the set $\mathbf{x} = \{x_j : j \in \mathbb{N}_m\}$ and $\mathbb{N}_m = \{1, \dots, m\}$. This functional trades off *empirical error*, measured by the function $Q : \mathbb{R}^m \rightarrow \mathbb{R}_+$, with *smoothness* of the solution, measured by the functional $\Omega : \mathcal{H} \rightarrow \mathbb{R}_+$. The empirical error depends upon a finite set $\{(x_j, y_j) : j \in \mathbb{N}_m\} \subset \mathcal{X} \times \mathbb{R}$ of *input-output examples* and the function Q depends on y but we suppress it in our notation since it is fixed throughout our discussion. In particular, one often considers the case that Q is defined, for $v = (v_j : j \in \mathbb{N}_m) \in \mathbb{R}^m$, as $Q(v) = \sum_{j \in \mathbb{N}_m} V(v_j, y_j)$ where $V : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_+$ is a prescribed *loss function*.

In this paper we assume that \mathcal{H} is a *reproducing kernel Hilbert space* (RKHS) \mathcal{H}_K with kernel K and choose $\Omega(f) = \langle f, f \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner product in \mathcal{H}_K , although some of the ideas we develop may be relevant in other circumstances. This leads us to study the variational problem

$$Q_{\mu}(K) := \inf \{ Q(I_{\mathbf{x}}(f)) + \mu \langle f, f \rangle : f \in \mathcal{H}_K \}. \quad (1)$$

We recall that an RKHS is a Hilbert space of real-valued functions everywhere defined on \mathcal{X} such that, for every $x \in \mathcal{X}$, the point evaluation functional defined, for $f \in \mathcal{H}$, by $L_x(f) := f(x)$ is continuous on \mathcal{H} (Aronszajn, 1950). This implies that \mathcal{H} admits a reproducing kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that, for every $x \in \mathcal{X}$, $K(x, \cdot) \in \mathcal{H}$ and $f(x) = \langle f, K(x, \cdot) \rangle$. In particular, for $x, t \in \mathcal{X}$, $K(x, t) = \langle K(x, \cdot), K(t, \cdot) \rangle$ implying that the $m \times m$ matrix $K_{\mathbf{x}} := (K(x_i, x_j) : i, j \in \mathbb{N}_m)$ is symmetric and positive semi-definite for *any* set of inputs $\mathbf{x} \subseteq \mathcal{X}$.

Often RKHS's are introduced through the notion of *feature map* $\Phi : \mathcal{X} \rightarrow \mathcal{W}$, where \mathcal{W} is a Hilbert space with inner product denoted by (\cdot, \cdot) . A feature map gives rise to the linear space of all functions $f : \mathcal{X} \rightarrow \mathbb{R}$ which are a linear combination of features whose norm is taken to be the norm of its coefficients. That is, for $w \in \mathcal{W}$, $f = (w, \Phi)$ and $\langle f, f \rangle = (w, w)$. This space is an RKHS with kernel K defined, for $x, t \in \mathcal{X}$, as $K(x, t) = (\Phi(x), \Phi(t))$. Using these equations, the regularization functional in (1) can be rewritten as a functional of w .

Regularization in an RKHS has a number of attractive features, including the availability of effective error bounds and stability analysis relative to perturbations of the data (see, for example, Bousquet and Elisseeff, 2002; Cucker and Smale, 2002; Mukherjee et al., in press; Scovel and Steinwart, 2004; Smale and Zhou, 2003; Vapnik, 1998; Ying and Zhou, 2004; Zhang, 2004; Zhou, 2002). Moreover, one can show that if f is a minimizer of the above functional it has the form

$$f(x) = \sum_{j \in \mathbb{N}_m} c_j K(x_j, x), \quad x \in \mathcal{X} \quad (2)$$

for some real vector $c = (c_j : j \in \mathbb{N}_m)$ of coefficients (see, for example, De Vito et al., 2005; Girosi, 1998; Kimeldorf and Wahba, 1971; Micchelli and Pontil, 2005; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004). This result is known in Machine Learning as the *representer theorem*. Although it is simple to prove, this result is remarkable as it makes the variational problem (1) amenable for computations.

If Q is convex, the unique minimizer of problem (1) can be found by replacing f by the right hand side of equation (2) in equation (1) and then optimizing with respect to the vector c . For example, when Q is the square error defined for $v = (v_j : j \in \mathbb{N}_m) \in \mathbb{R}^m$ as $Q(v) = \sum_{j \in \mathbb{N}_m} (v_j - y_j)^2$ the functional in the right hand side of (1) is a quadratic in the vector c and its minimizer is obtained by solving a linear system of equations.

Because of their simplicity and generality, kernels and associated RKHS's play an increasingly important role in Machine Learning, Pattern Recognition and their applications. This was initiated with the introduction of support vector machines (see, for example, Vapnik, 1998), and continued with the development of many other kernel-based learning algorithms (see, for example, Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004, and references therein). As kernels can be defined on any input space, kernel-based methods have been successfully applied to learning functions defined on complex data structures, ranging from images, text data, speech data, biological data, among others.

Despite this great success, there still remain important problems to be addressed concerning kernel methods in Machine Learning. When the kernel is fixed, an immediate concern with problem (1) is the *choice of the regularization parameter* μ . This is typically solved by means of cross validation or generalized cross validation (see, for example, Hastie, Tibshirani and Friedman, 2002; Wahba, 1990) or by means of regularization path methods (see, for example, Bach, Thibaux and Jordan, 2004; Hastie et al., 2004; Pontil and Verri, 1998). But, how is the kernel chosen? Indeed, a challenging and central problem is the *choice of the kernel* itself. As we said before, when \mathcal{H}

is constructed as linear combinations of features associated to the kernel K , they can provide some guideline for the choice of the kernel. Thus, the choice of the kernel is tied to the problem of choosing a representation of the input. This choice can make a significant difference in practice. For example, techniques such as radial basis functions can perform poorly if the parameter of the radial kernel is not tuned to the given data. A similar circumstance occurs for translation invariant kernels modeled by Gaussian mixtures. When the number of parameters is large cross validation encounters severe computational limitations. To overcome this problem, easily computable approximations to the leave-one-out error have been derived (Chapelle et al., 2002; Wahba, 1990). Nonetheless, these methods are usually non-convex and may lead to undesirable local minima.

In this paper, we propose a method for finding a kernel function which belongs to a *compact* and *convex* set \mathcal{K} . Our method is based on the minimization of the functional in equation (1), that is, we consider the variational problem

$$\inf\{Q_\mu(K) : K \in \mathcal{K}\}. \quad (3)$$

This problem shares some similarities with recent progress in the context of kernel-based methods (Bach, Lanckriet and Jordan, 2004; Bousquet and Herrmann, 2003; Cristianini et al., 2002; Graepel, 2002; Lanckriet et al., 2002, 2004; Lee et al., 2004; Lin and Zhang, 2003; Herbster, 2001; Ong, Smola and Williamson, 2003; Wu, Ying and Zhou, 2004; Zhang, Yeung and Kwok, 2004). In particular, the third and fifth papers motivated our work. In contrast to the point of view of these papers, our setting applies to convex combinations of kernels parameterized by a compact set, a circumstance which is relevant for applications. We also wish to emphasize that although we focus on learning methods based on the minimization of the functional (1), the ideas which we present here may prove useful for learning kernels or feature representations using different forms of regularization such as entropy regularization (Jaakkola, Meila and Jebara, 1999), kernel density estimation (see, for example, Vapnik, 1998), or one-class SVM (Tax and Duin, 1999) as well as in other Machine Learning frameworks such as those arising in Bayesian learning where a kernel is seen as the covariance of a Gaussian process, (see, for example, Wahba, 1990; Williams and Rasmussen, 1996) or in online learning, (see, for example, Herbster, 2001).

In Section 2 we establish the existence of a solution to problem (3), show that the functional Q_μ is *convex* in K , and observe that, although \mathcal{K} may be an uncountable set, the optimal kernel is always obtained as a convex combination of at most $m+2$ basic kernels (see below), where m is the number of training data. The simplest case of our setup is a set of convex combinations of finitely many kernels $\{K_j : j \in \mathbb{N}_n\}$. For example each K_j could be a Gaussian, a polynomial kernel, or simply a kernel consisting of only one feature. In all of these cases our method will seek the optimal convex combination of these kernels. Another example included in our framework is learning the optimal radial kernel or the optimal polynomial kernel in which case the space \mathcal{K} is the convex hull of a prescribed set of kernels parameterized by a *locally compact* set. In Section 3 we study square loss regularization and provide improvements and simplifications of the results in Section 2. In particular, we discuss the connection to minimal norm interpolation and establish necessary and sufficient conditions for a kernel to be optimal. Finally, in Section 4 we comment on previous work, present some numerical simulations based on our analysis and discuss some extensions of our framework.

2. Optimal Convex Combination of Kernels

Let \mathcal{X} be a set. By a *kernel* we mean a symmetric function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that for every finite set of inputs $\mathbf{x} = \{x_j : j \in \mathbb{N}_m\} \subseteq \mathcal{X}$ and every $m \in \mathbb{N}$, the $m \times m$ matrix $K_{\mathbf{x}} := (K(x_i, x_j) : i, j \in \mathbb{N}_m)$ is positive semi-definite. We let $\mathcal{L}(\mathbb{R}^m)$ be the set of $m \times m$ positive semi-definite matrices and $\mathcal{L}_+(\mathbb{R}^m)$ the subset of positive definite ones. Also, we use $\mathcal{A}(\mathcal{X})$ for the set of all kernels on the set \mathcal{X} and $\mathcal{A}_+(\mathcal{X})$ for the subset of kernels K such that, for each input \mathbf{x} , $K_{\mathbf{x}} \in \mathcal{L}_+(\mathbb{R}^m)$. We also occasionally refer to the set of *all* symmetric $m \times m$ matrices and use $\mathcal{S}(\mathbb{R}^m)$ to denote them.

According to Aronszajn and Moore (see Aronszajn, 1950), every kernel has associated to it an (essentially) *unique* Hilbert space \mathcal{H}_K with inner product $\langle \cdot, \cdot \rangle$ such that K is its reproducing kernel. This means that for every $f \in \mathcal{H}_K$ and $x \in \mathcal{X}$, $\langle f, K_x \rangle = f(x)$, where K_x is the function $K(x, \cdot)$.

Let $D := \{(x_j, y_j) : j \in \mathbb{N}_m\} \subset \mathcal{X} \times \mathbb{R}$ be prescribed data and y the vector $(y_j : j \in \mathbb{N}_m)$. For each $f \in \mathcal{H}_K$, we introduce the *information operator* $I_{\mathbf{x}}(f) := (f(x_j) : j \in \mathbb{N}_m)$ of values of f on the set $\mathbf{x} := \{x_j : j \in \mathbb{N}_m\}$. We prescribe a nonnegative function $Q : \mathbb{R}^m \rightarrow \mathbb{R}_+$ and introduce the *regularization functional*

$$Q_{\mu}(f, K) := Q(I_{\mathbf{x}}(f)) + \mu \|f\|_K^2 \tag{4}$$

where $\|f\|_K^2 := \langle f, f \rangle$, μ is a positive constant and Q depends on y but we suppress it in our notation as it is fixed throughout our discussion. A noteworthy special case of Q_{μ} is the *square loss regularization functional* given by

$$S_{\mu}(f, K) := \|y - I_{\mathbf{x}}(f)\|^2 + \mu \|f\|_K^2 \tag{5}$$

where $\|\cdot\|$ is the standard Euclidean norm on \mathbb{R}^m . There are many other choices of the functional Q_{μ} which are important for applications, see the work of Vapnik (1998) for a discussion.

Associated with the functional Q_{μ} and the kernel K is the variational problem

$$Q_{\mu}(K) := \inf\{Q_{\mu}(f, K) : f \in \mathcal{H}_K\} \tag{6}$$

which defines a function $Q_{\mu} : \mathcal{A}(\mathcal{X}) \rightarrow \mathbb{R}_+$. We remark, in passing, that all of what we say about problem (6) applies to functions Q which are bounded from below on \mathbb{R}^m as we can merely adjust the expression (4) by a constant independent of f and K . Let us first point out that the infimum in (6) is achieved, at least when Q is continuous.

Lemma 1 *If $Q : \mathbb{R}^m \rightarrow \mathbb{R}_+$ is continuous and μ is a positive number then the infimum in (6) is achieved for a function in \mathcal{H}_K . Moreover, when Q is convex this function is unique.*

PROOF. The proof of this fact is straightforward and uses *weak compactness* of the unit ball in \mathcal{H}_K . The uniqueness of the solution relies on the fact that when Q is convex Q_{μ} is strictly convex because μ is positive. □

The point of view of this paper is that the functional (6) can be used as a *design criterion to select the kernel K* . To this end, we specify an arbitrary convex subset \mathcal{K} of $\mathcal{A}(\mathcal{X})$ and focus on the problem

$$Q_{\mu}(\mathcal{K}) = \inf\{Q_{\mu}(K) : K \in \mathcal{K}\}. \tag{7}$$

Recall that the solution of (6) is given in the form $f = \sum_{j \in \mathbb{N}_m} c_j K_{x_j}$ for some vector $c := (c_j : j \in \mathbb{N}_m)$, (see, for example, De Vito et al., 2005; Girosi, 1998; Kimeldorf and Wahba, 1971; Micchelli and Pontil, 2005; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004). Such a function

representation for learning the function f is central for many diverse applications of kernel-based algorithms in Machine Learning. Moreover, the coefficient vector c is found as the solution of the *finite dimensional* variational problem

$$Q_\mu(K) := \min\{Q(K_{\mathbf{x}}c) + \mu(c, K_{\mathbf{x}}c) : c \in \mathbb{R}^m\}$$

where (\cdot, \cdot) is the standard inner product on \mathbb{R}^m .

Before we address basic questions concerning the variational problem (7) we describe some terminology that allows for a precise description of our observations. Every input set \mathbf{x} and set of *basic kernels* \mathcal{G} on $\mathcal{X} \times \mathcal{X}$ determines a set of *matrices* in $\mathcal{L}(\mathbb{R}^m)$, namely

$$\mathcal{G}(\mathbf{x}) := \{G_{\mathbf{x}} : G \in \mathcal{G}\}.$$

Obviously, it is the set of matrices $\mathcal{K}(\mathbf{x})$ that affects the variational problem (7). Note that $\mathcal{G}(\mathbf{x})$ being a subset of $\mathcal{S}(\mathbb{R}^m)$ is identifiable as a set of vectors in \mathbb{R}^N , where $N := \frac{m(m+1)}{2}$. As such $\mathcal{G}(\mathbf{x})$ inherits the standard topology from \mathbb{R}^N . That is, convergence of a sequence of matrices in $\mathcal{G}(\mathbf{x})$ means that the respective elements of the matrices converge. For this reason, we use $\overline{\mathcal{G}}$ (the closure of \mathcal{G}) to mean the set of all kernels K on $\mathcal{X} \times \mathcal{X}$ with the property that for each $\mathbf{x} \subseteq \mathcal{X}$, the matrix $K_{\mathbf{x}} \in \overline{\mathcal{G}(\mathbf{x})}$, the closure of $\mathcal{G}(\mathbf{x})$ relative to \mathbb{R}^N . We say a set of kernels \mathcal{G} is closed provided that $\overline{\mathcal{G}} = \mathcal{G}$. Also, we say \mathcal{G} is a compact convex set of kernels whenever for each $\mathbf{x} \subseteq \mathcal{X}$, $\mathcal{G}(\mathbf{x})$ is a compact convex set of matrices in $\mathcal{S}(\mathbb{R}^m)$. Our next result establishes the existence of the solution to problem (7).

Lemma 2 *If \mathcal{K} is a compact and convex subset of $\mathcal{A}_+(\mathcal{X})$ and $Q : \mathbb{R}^m \rightarrow \mathbb{R}$ is continuous then the minimum of (7) exists.*

PROOF. Fix $\mathbf{x} \subseteq \mathcal{X}$, choose a minimizing sequence of kernels $\{K^n : n \in \mathbb{N}\}$, that is, $\lim_{n \rightarrow \infty} Q_\mu(K^n) = Q_\mu(\mathcal{K})$ and a sequence of vectors $\{c^n : n \in \mathbb{N}\}$ such that

$$Q_\mu(K^n) = Q(K_{\mathbf{x}}^n c^n) + \mu(c^n, K_{\mathbf{x}}^n c^n).$$

Since \mathcal{K} is compact there is a subsequence $\{K^{n(\ell)} : \ell \in \mathbb{N}\}$ such that $\lim_{\ell \rightarrow \infty} K_{\mathbf{x}}^{n(\ell)} = \tilde{K}_{\mathbf{x}}$, for some kernel $\tilde{K} \in \mathcal{K}$. We claim that $\{c^n : n \in \mathbb{N}\}$ is bounded. Indeed, there is a positive constant ρ such that $(c^n, K_{\mathbf{x}}^n c^n) \leq \rho$. Set $a^n = \frac{c^n}{\|c^n\|}$ so that $(a^n, K_{\mathbf{x}}^n a^n) \leq \frac{\rho}{\|c^n\|^2}$ and choose a convergent subsequence $\{a^{n(\ell(q))} : q \in \mathbb{N}\}$ such that $\lim_{q \rightarrow \infty} a^{n(\ell(q))} = a$ and $\|a\| = 1$ for some vector $a \in \mathbb{R}^m$. If the sequence $\{c^n : n \in \mathbb{N}\}$ is not bounded we conclude that $(a, \tilde{K}_{\mathbf{x}} a) = 0$ contradicting our hypothesis that $\tilde{K} \in \mathcal{A}_+(\mathcal{X})$. Hence there is a subsequence $\{c^{n(\ell(q))} : q \in \mathbb{N}\}$ such that $\lim_{q \rightarrow \infty} c^{n(\ell(q))} = c$, for some $c \in \mathbb{R}^m$. Therefore, we conclude that

$$Q_\mu(\mathcal{K}) = Q(\tilde{K}_{\mathbf{x}} c) + \mu(c, \tilde{K}_{\mathbf{x}} c) \geq Q_\mu(\tilde{K})$$

from which it follows that $Q_\mu(\mathcal{K}) = Q_\mu(\tilde{K})$. □

The proof of this lemma requires that all kernels in \mathcal{K} are in $\mathcal{A}_+(\mathcal{X})$. If we wish to use kernels K only in $\mathcal{A}(\mathcal{X})$ we may always modify them by adding *any* positive multiple of the *delta function kernel* Δ defined, for $x, t \in \mathcal{X}$, as

$$\Delta(x, t) = \begin{cases} 1, & x = t \\ 0, & x \neq t \end{cases} \quad (8)$$

that is, replace K by $K + a\Delta$ where a is a positive constant.

There are two useful cases of a set \mathcal{K} of kernels which are compact and convex. The first is formed by the convex hull of a *finite* number of kernels in $\mathcal{A}_+(\mathcal{X})$. The second example extends this to a compact Hausdorff space Ω , (see, for example, Royden, 1988), and a mapping $G : \Omega \rightarrow \mathcal{A}_+(\mathcal{X})$. For each $\omega \in \Omega$, the value of the kernel $G(\omega)$ at $x, t \in \mathcal{X}$ is denoted by $G(\omega)(x, t)$ and we assume that the function of $\omega \mapsto G(\omega)(x, t)$ is continuous on Ω for each $x, t \in \mathcal{X}$. When this is the case we say G is *continuous*. We let $\mathcal{M}(\Omega)$ be the set of all *probability measures* on Ω and observe that

$$\mathcal{K}(G) := \left\{ \int_{\Omega} G(\omega) dp(\omega) : p \in \mathcal{M}(\Omega) \right\} \tag{9}$$

is a compact and convex set of kernels in $\mathcal{A}_+(\mathcal{X})$. The compactness of the set $\mathcal{K}(G)$ is a consequence of weak*-compactness of the unit ball of the dual space of $C(\Omega)$, the set of all continuous real-valued functions g on Ω with norm $\|g\|_{\Omega} := \max\{|g(\omega)| : \omega \in \Omega\}$ (Royden, 1988). For example, we choose $\Omega = [a, b]$, where $a > 0$ and $G(\omega)(x, t) = e^{-\omega\|x-t\|^2}$, $x, t \in \mathbb{R}^d$, $\omega \in \Omega$, to obtain *radial kernels*, or $G(\omega)(x, t) = e^{\omega(x,t)}$, $x, t \in \mathbb{R}^d$ to obtain *dot product kernels*. Note that the choice $\Omega = \mathbb{N}_n$ corresponds to our first example.

In preparation for the next theorem we need to express the set $\mathcal{K}(G)$ in an alternate form. We have in mind the following basic fact.

Lemma 3 *If Ω is a compact Hausdorff space, $G : \Omega \rightarrow \mathcal{A}_+(\mathcal{X})$ a continuous map as defined above and $\mathcal{G} := \{G(\omega) : \omega \in \Omega\}$ then $\mathcal{K}(G) = \overline{co\mathcal{G}}$.*

PROOF. First, we shall show that $\overline{co\mathcal{G}} \subseteq \mathcal{K}(G)$. To this end, we let $K \in \overline{co\mathcal{G}}$ and $\mathbf{x} \subseteq \mathcal{X}$. By the definition of convex hull, we obtain, for some sequence of probability measures $\{p_{\ell} : \ell \in \mathbb{N}\}$, that $K_{\mathbf{x}} = \lim_{\ell \rightarrow \infty} \int_{\Omega} G_{\mathbf{x}}(\omega) dp_{\ell}(\omega)$ where each p_{ℓ} is a *finite* sum of point measures. Since for each $\ell \in \mathbb{N}$, $\int_{\Omega} G_{\mathbf{x}}(\omega) dp_{\ell}(\omega) \in \mathcal{K}(G)$ and $\mathcal{K}(G)$ is closed it follows that $K_{\mathbf{x}} \in \mathcal{K}(G)$, that is, we have established that $\overline{co\mathcal{G}} \subseteq \mathcal{K}(G)$.

On the other hand, if there is a kernel $K \in \mathcal{K}(G)$ which does not belong to $\overline{co\mathcal{G}}$ then there is an input set \mathbf{x} such that $K_{\mathbf{x}} \notin \overline{co\mathcal{G}(\mathbf{x})}$ while $K_{\mathbf{x}} = \int_{\Omega} G_{\mathbf{x}}(\omega) dp(\omega)$ for some $p \in \mathcal{M}(\Omega)$. Hence, there exists a hyperplane which separates the matrix $K_{\mathbf{x}}$ from the set of matrices $\overline{co\mathcal{G}(\mathbf{x})}$ (Royden, 1988). This means that there is a linear functional L on $\mathcal{S}(\mathbb{R}^m)$ and $c \in \mathbb{R}$ such that $L(K_{\mathbf{x}}) > c$ but $L(G_{\mathbf{x}}(\omega)) < c$ for all $\omega \in \Omega$. We integrate the last inequality over $\omega \in \Omega$ relative to the measure dp and conclude by the linearity of L that $L(K_{\mathbf{x}}) < c$, a contradiction. This concludes the proof. \square

Observe that the set $\mathcal{G} = \{G(\omega) : \omega \in \Omega\}$ in the above lemma is compact since G is continuous and Ω compact. In general, we wish to point out a useful fact about the kernels in $\overline{co\mathcal{G}}$ whenever \mathcal{G} is a *compact* set of kernels. To this end, we recall a theorem of Caratheodory (see, for example, Rockafellar, 1970, Ch. 17).

Theorem 4 *If A is a subset of \mathbb{R}^n then every $a \in coA$ is a convex combination of at most $n + 1$ elements of A .*

An immediate consequence of the above theorem is the following fact which we shall use later.

Lemma 5 *If A is a compact subset of \mathbb{R}^n then \overline{coA} is compact and every element in it is a convex combination of at most $n + 1$ elements of A .*

In particular, we have the following corollary.

Corollary 6 *If \mathcal{G} is a compact set of kernels on $X \times X$ then $\overline{co\mathcal{G}}$ is a compact set of kernels. Moreover, for each input set \mathbf{x} a matrix $C \in \overline{co\mathcal{G}}(\mathbf{x})$ if and only if there exists a kernel T which is a convex combination of at most $\frac{m(m+1)}{2} + 1$ kernels in \mathcal{G} and $T_{\mathbf{x}} = C$.*

Our next result shows whenever \mathcal{K} is the closed convex hull of a compact set of kernels \mathcal{G} that the optimal kernel lies in a the convex hull of some *finite* subset of \mathcal{G} .

Theorem 7 *If $\mathcal{G} \subseteq \mathcal{A}_+(X)$ is a compact set of basic kernels, $\mathcal{K} = \overline{co\mathcal{G}}$, $Q : \mathbb{R}^m \rightarrow \mathbb{R}_+$ is continuous and μ is a positive number then there exists $\mathcal{T} \subseteq \mathcal{G}$ containing at most $m + 2$ basic kernels such that Q_μ admit a minimizer $\tilde{K} \in co\mathcal{T}$ and $Q_\mu(\mathcal{T}) = Q_\mu(\mathcal{K})$.*

PROOF. Let $(\hat{c}, \hat{K}) \in \mathbb{R}^m \times \mathcal{K}$ be a minimizer of Q_μ , that is, we have that

$$Q_\mu(\mathcal{K}) = \min\{Q(\hat{K}_{\mathbf{x}}c) + \mu(c, \hat{K}_{\mathbf{x}}c) : c \in \mathbb{R}^m\} = Q(\hat{K}_{\mathbf{x}}\hat{c}) + \mu(\hat{c}, \hat{K}_{\mathbf{x}}\hat{c}).$$

We define the set of vectors $\mathcal{U} := \{(K_{\mathbf{x}}\hat{c}, (\hat{c}, K_{\mathbf{x}}\hat{c})) : K \in \mathcal{K}\} \subset \mathbb{R}^{m+1}$. Note that $\mathcal{U} = \overline{co\mathcal{V}}$ where $\mathcal{V} = \{(G_{\mathbf{x}}\hat{c}, (\hat{c}, G_{\mathbf{x}}\hat{c})) : G \in \mathcal{G}\}$ and \mathcal{V} is compact since \mathcal{G} is compact. By Lemma 5 the vector $(\hat{K}_{\mathbf{x}}\hat{c}, (\hat{c}, \hat{K}_{\mathbf{x}}\hat{c}))$ can be written as a convex combination of at most $m + 2$ vectors in \mathcal{V} , that is

$$(\hat{K}_{\mathbf{x}}\hat{c}, (\hat{c}, \hat{K}_{\mathbf{x}}\hat{c})) = (\tilde{K}_{\mathbf{x}}\hat{c}, (\hat{c}, \tilde{K}_{\mathbf{x}}\hat{c}))$$

where \tilde{K} is the convex combination of at most $m + 2$ kernels in \mathcal{G} . Consequently, we have that

$$\begin{aligned} Q_\mu(\mathcal{K}) &= Q(\tilde{K}_{\mathbf{x}}\hat{c}) + \mu(\hat{c}, \tilde{K}_{\mathbf{x}}\hat{c}) \\ &\geq \min\{Q(\tilde{K}_{\mathbf{x}}c) + \mu(c, \tilde{K}_{\mathbf{x}}c) : c \in \mathbb{R}^m\} \\ &= Q_\mu(\tilde{K}) \geq Q_\mu(\mathcal{K}) \end{aligned}$$

implying that $Q_\mu(\hat{K}) = Q_\mu(\tilde{K})$. □

Note that Theorem 7 asserts the *existence* of a q which is *at most* $m + 2$, that is, an optimal kernel is expressed by a convex combination of at most $m + 2$ kernels.

Note that in the definition of $Q_\mu(\mathcal{K})$ we minimize first over $f \in \mathcal{H}_K$ and then over $K \in \mathcal{K}$. There arises the question of what would happen if we interchange these minima. We address this issue in the case that \mathcal{K} is the convex hull of a finite set of kernels. To this end, we use the notation $\bigoplus_{j \in \mathbb{N}_n} \mathcal{H}_{K_j}$ for the direct sum of the Hilbert spaces $\{\mathcal{H}_{K_j} : j \in \mathbb{N}_n\}$.

Lemma 8 *If $\mathcal{K}_u = \{K_j : j \in \mathbb{N}_n\}$ is a family of kernels on $X \times X$ and $f \in \bigoplus_{j \in \mathbb{N}_n} \mathcal{H}_{K_j}$ then*

$$\inf\{\|f\|_K : K \in co\mathcal{K}_u\} = \min \left\{ \sum_{j \in \mathbb{N}_n} \|f_j\|_{K_j} : f = \sum_{j \in \mathbb{N}_n} f_j, f_\ell \in \mathcal{H}_{K_\ell}, \ell \in \mathbb{N}_n \right\}. \quad (10)$$

As the result is not needed in our subsequent analysis we postpone its proof to the appendix (for related results, see also, Herbster, 2004; Lin and Zhang, 2003). We note that the expression on the right hand side of equation (10) is an *intermediate* norm for $\bigoplus_{j \in \mathbb{N}_n} \mathcal{H}_{K_j}$ (see Bennett and Sharpley, 1988, p. 97) for a discussion. This lemma suggests a reformulation of our extremal problem (7) for kernels of the form (9) where G is expressed in terms of a feature map. Although this fact is interesting, it is not central to our point of view in this paper and, so, we describe it in the appendix.

Next, we establish that the variational problem (7) is a *convex optimization problem*. Specifically, we shall show that if the function mapping $Q : \mathbb{R}^m \rightarrow \mathbb{R}$ is convex then the functional $Q_\mu : \mathcal{A}_+(\mathcal{X}) \rightarrow \mathbb{R}_+$ is a convex as well. It is curious that this does not seem to follow directly from the *definition* of Q_μ . We take a sojourn through the notion of *conjugate function*. Recall that the conjugate function of Q denoted by $Q^* : \mathbb{R}^m \rightarrow \mathbb{R}$ is defined, for every $v \in \mathbb{R}^m$, as

$$Q^*(v) = \sup\{(c, v) - Q(c) : c \in \mathbb{R}^m\}$$

and it follows, for every $c \in \mathbb{R}^m$, that

$$Q(c) = \sup\{(c, v) - Q^*(v) : v \in \mathbb{R}^m\}$$

(see, for example, Rockafellar, 1970; Borwein and Lewis, 2000). A nice recent application of the conjugate function to linear statistical models appears in (Zhang, 2002).

The proof we present below for the convexity of $Q_\mu : \mathcal{A}_+(\mathcal{X}) \rightarrow \mathbb{R}_+$ is based upon the von Neumann minimax theorem which we record in the appendix. We begin by introducing for each $r > 0$ a function $\phi_r : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ defined, for $t \in \mathbb{R}_+$, as

$$\phi_r(t) := \mu \left(\frac{1}{2\mu} \sqrt{t} - r \right)_+^2 - \frac{1}{4\mu} t$$

where $(z)_+ := \max(0, z)$. Note that

$$\lim_{r \rightarrow \infty} \phi_r(t) = -\frac{1}{4\mu} t$$

pointwise for $t > 0$. Also, for each fixed $t > 0$, $\phi_r(t)$ is a non-increasing function of r and, for each $r > 0$, ϕ_r is continuously differentiable, decreasing and convex on \mathbb{R}_+ .

Lemma 9 *If $K \in \mathcal{A}(\mathcal{X})$, \mathbf{x} a set of m distinct points of \mathcal{X} such that $K_{\mathbf{x}} \in \mathcal{L}_+(\mathbb{R}^m)$ and $Q : \mathbb{R}^m \rightarrow \mathbb{R}$ a convex function, then there exists $r_0 > 0$ such that for all $r > r_0$ there holds the formula*

$$Q_\mu(K) = \sup\{\phi_r((v, K_{\mathbf{x}}v)) - Q^*(v) : v \in \mathbb{R}^m\}. \tag{11}$$

PROOF. By the definition of Q_μ we have that

$$Q_\mu(K) = \min\{\sup\{(K_{\mathbf{x}}c, v) - Q^*(v) + \mu(c, K_{\mathbf{x}}c) : v \in \mathbb{R}^m\} : c \in \mathbb{R}^m\}.$$

According to Lemma 2 the minimum above exists. Therefore, there is a $r_0 > 0$ such that for all $r > r_0$ we have that

$$Q_\mu(K) = \min\{\sup\{(K_{\mathbf{x}}c, v) - Q^*(v) + \mu(c, K_{\mathbf{x}}c) : v \in \mathbb{R}^m\} : c \in \mathbb{R}^m, (c, K_{\mathbf{x}}c) \leq r^2\}.$$

By the minimax theorem, see Theorem 22 in the appendix, we conclude that

$$Q_\mu(K) = \sup\{\min\{(K_{\mathbf{x}}c, v) - Q^*(v) + \mu(c, K_{\mathbf{x}}c) : c \in \mathbb{R}^m, (c, K_{\mathbf{x}}c) \leq r^2\} : v \in \mathbb{R}^m\}.$$

For each $v \in \mathbb{R}^m$, we shall now explicitly compute the minimum of the above expression. To this end, we let $K_{\mathbf{x}} := B^2$ where B is a $m \times m$ positive definite matrix, that is, B is the square root of $K_{\mathbf{x}}$, and observe that

$$\min\{(c, K_{\mathbf{x}}v) + \mu(c, K_{\mathbf{x}}c) : (c, K_{\mathbf{x}}c) \leq r^2\} = \min\{\mu\|Bc + \frac{1}{2\mu}Bv\|^2 - \frac{1}{4\mu}\|Bv\|^2 : \|Bc\| \leq r\}.$$

If the vector $c_0 := -\frac{1}{2\mu}v$ has the property that $\|Bc_0\| \leq r$, that is, $\|Bv\| \leq 2\mu r$ then the minimum above is $-\frac{1}{4\mu}\|Bv\|^2$, otherwise $\|Bv\| > 2\mu r$ and the triangle inequality says that

$$\|Bc + \frac{1}{2\mu}Bv\| \geq \frac{1}{2\mu}\|Bv\| - \|Bc\| \geq \frac{1}{2\mu}\|Bv\| - r.$$

Since, for the vector $\hat{c} := -\frac{v}{\|Bv\|}$, we have that

$$\|B\hat{c} + \frac{1}{2\mu}Bv\| = \frac{1}{2\mu}\|Bv\| - r$$

this inequality is sharp. Therefore, we get that

$$Q_\mu(K) = \sup\left\{\mu\left(\frac{1}{2\mu}\|Bv\| - r\right)_+^2 - \frac{1}{4\mu}\|Bv\|^2 - Q^*(v) : v \in \mathbb{R}^m\right\}$$

and the result follows by the definition of ϕ_r . \square

Let us specialize this lemma to the example of the square loss S defined, for $w \in \mathbb{R}^m$, as $S(w) = \|y - w\|^2$. In this case, the conjugate function is given explicitly for $v \in \mathbb{R}^m$ as

$$S^*(v) = \max\{(w, v) - \|w - y\|^2 : w \in \mathbb{R}^m\} = \frac{1}{4}\|v\|^2 + (y, v).$$

We shall show later in Lemma 14 by a *direct* computation *without* the use of the conjugate function that $S_\mu = \mu(y, (K_{\mathbf{x}} + \mu I)^{-1}y)$. Alternatively, if we formally let $r = \infty$ in the right hand side of equation (11) we get

$$\sup\left\{-\frac{1}{4\mu}(v, (K_{\mathbf{x}} + \mu I)v) - (y, v) : v \in \mathbb{R}^m\right\}$$

which by a direct computation equals $\mu(y, (K_{\mathbf{x}} + \mu I)^{-1}y)$. This suggests that Lemma 9 may even hold when $r = \infty$ and without the hypothesis that $K_{\mathbf{x}} \in \mathcal{L}_+(\mathbb{R}^m)$. We shall confirm this with another version of the von Neumann minimax theorem.

Lemma 10 *If $K \in \mathcal{A}(X)$, \mathbf{x} a set of m distinct points of X such that $K_{\mathbf{x}} \in \mathcal{L}_+(\mathbb{R}^m)$ and $Q : \mathbb{R}^m \rightarrow \mathbb{R}$ a convex function, then there holds the formula*

$$Q_\mu(K) = \sup\left\{-\frac{1}{4\mu}(v, K_{\mathbf{x}}v) - Q^*(v) : v \in \mathbb{R}^m\right\}.$$

PROOF. Theorem 23 applies since $K_{\mathbf{x}} \in \mathcal{L}_+(\mathbb{R}^m)$. Indeed, we let $f(c, v) = (K_{\mathbf{x}}c, v) - Q^*(v) + \mu(c, K_{\mathbf{x}}c)$, $\mathcal{A} = \mathcal{B} = \mathbb{R}^m$ and $v_0 = 0$ then the set $\{c : c \in \mathbb{R}^m, f(c, v_0) \leq \lambda\}$ is compact and all the hypotheses of Theorem 23 hold. Hence, we may proceed as in the proof of Lemma 9 with $r = \infty$. \square

To interpret Lemma 9, we say that $A \preceq B$ whenever $A, B \in \mathcal{L}(\mathbb{R}^m)$, if $B - A$ is positive semi-definite. We also say, for $K, J \in \mathcal{A}(\mathcal{X})$, that $K \preceq J$ if $K_{\mathbf{x}} \preceq J_{\mathbf{x}}$ for every $\mathbf{x} \subseteq \mathcal{X}$.

Definition 11 A function $\phi : \mathcal{B} \rightarrow \mathbb{R}$ is said non-decreasing on $\mathcal{B} \subseteq \mathcal{A}(\mathcal{X})$ if, for every $A, B \in \mathcal{B}$ with $A \preceq B$ it follows that $\phi(A) \leq \phi(B)$. If the reverse inequality holds we say ϕ is non-increasing.

Definition 12 A function $\phi : \mathcal{B} \rightarrow \mathbb{R}$ is said convex on $\mathcal{B} \subseteq \mathcal{A}(\mathcal{X})$ if, for every $A, B \in \mathcal{B}$ and $\lambda \in [0, 1]$ there holds the inequality

$$\phi(\lambda A + (1 - \lambda)B) \leq \lambda\phi(A) + (1 - \lambda)\phi(B). \quad (12)$$

If the reverse of inequality (12) holds we say that the ϕ is concave.

Proposition 13 If $Q : \mathbb{R}^m \rightarrow \mathbb{R}_+$ is convex then for every $\mu > 0$ the function $Q_{\mu} : \mathcal{A}_+(\mathcal{X}) \rightarrow \mathbb{R}_+$ is convex and non-increasing.

PROOF. The proof of the proposition follows from Lemma 9. Specifically, equation (11) expresses Q_{μ} as the supremum of a family of functions which are convex and non-increasing on $\mathcal{A}(\mathcal{X})$. \square

We note that the convexity of the function Q_{μ} was already proven by Lanckriet et al. (2004) for the hinge loss and stated in (Ong, Smola and Williamson, 2003) for differentiable convex loss functions.

3. Square Regularization

In this section we exclusively study the case of the square loss regularization functional S_{μ} in equation (5) and provide improvements and simplifications of our previous results. We begin by determining the *explicit* expression for this functional which we briefly mentioned earlier after the proof of Lemma 9.

Lemma 14 For any kernel K , inputs $\mathbf{x} := \{x_j : j \in \mathbb{N}_m\}$, samples $y = (y_j : j \in \mathbb{N}_m)$ and positive constant μ we have that

$$S_{\mu}(K) = \mu(y, (\mu I + K_{\mathbf{x}})^{-1}y) \quad (13)$$

where I is the $m \times m$ identity matrix.

PROOF. We have that $S_{\mu}(K) = \min\{R(c) : c \in \mathbb{R}^m\}$ where for each $c \in \mathbb{R}^m$ we set $R(c) := \|y - K_{\mathbf{x}}c\|^2 + \mu(c, K_{\mathbf{x}}c)$. We define the vector $w := (\mu I + K_{\mathbf{x}})^{-1}y$, observe that $R(w) = (y, \mu(\mu I + K_{\mathbf{x}})^{-1}y)$ and for every vector $c \in \mathbb{R}^m$ we have that

$$R(c) = R(w) + \|K_{\mathbf{x}}(w - c)\|^2 + \mu(c - w, K_{\mathbf{x}}(c - w)).$$

With this formula the result follows. \square

From this lemma we conclude, when the matrix $K_{\mathbf{x}}$ is in $\mathcal{L}_+(\mathbb{R}^m)$ then $\lim_{\mu \rightarrow 0} \mu^{-1} S_{\mu}(K) = \gamma(K_{\mathbf{x}})$, where for every $A \in \mathcal{L}_+(\mathbb{R}^m)$ we set $\gamma(A) := (y, A^{-1}y)$. The function $\gamma: \mathcal{L}_+(\mathbb{R}^m) \rightarrow \mathbb{R}_+$ has the alternate form

$$\frac{1}{\gamma(A)} := \min\{(c, Ac) : c \in \mathbb{R}^m, (c, y) = 1\}, \quad A \in \mathcal{L}_+(\mathbb{R}^m) \quad (14)$$

and the unique vector which achieves this minimum is given by

$$c(A) := \frac{A^{-1}y}{(y, A^{-1}y)}. \quad (15)$$

A proof of these facts follow directly from the Cauchy-Schwarz inequality for the inner product (u, Av) , $u, v \in \mathbb{R}^m$. Moreover, this alternate form for $\gamma(A)$ connects the function γ to the *minimal norm interpolant* in \mathcal{H}_K to the data D . Let us explain this connection next.

Recall, for every kernel K on $\mathcal{X} \times \mathcal{X}$, that the minimal norm interpolation to the data D is the solution to the variational problem

$$\rho(K) := \min\{\|f\|_K^2 : f \in \mathcal{H}_K, f(x_j) = y_j, j \in \mathbb{N}_m\}. \quad (16)$$

The following result is well-known (for a proof see, for example, Micchelli and Pontil, 2005).

Proposition 15 *If $K \in \mathcal{A}(\mathcal{X})$ and \mathbf{x} is an input set in \mathcal{X} such that the matrix $K_{\mathbf{x}}$ is in $\mathcal{L}_+(\mathbb{R}^m)$ then the solution of the minimal norm interpolation problem (16) is unique and is given by*

$$f = \sum_{j \in \mathbb{N}_m} c_j K(x_j, \cdot)$$

where the coefficient vector $c = (c_j : j \in \mathbb{N}_m)$ solves the linear system of equations $K_{\mathbf{x}}c = y$ and we have that

$$\rho(K) = \gamma(K_{\mathbf{x}}) = (y, K_{\mathbf{x}}^{-1}y). \quad (17)$$

The function $\gamma: \mathcal{L}_+(\mathbb{R}^m) \rightarrow \mathbb{R}_+$ is continuous. We record additional facts about this function in the next two lemmas.

Lemma 16 *The function γ is non-increasing and whenever $A, B \in \mathcal{L}_+(\mathbb{R}^m)$, $\gamma(A) = \gamma(B)$ if and only if $A^{-1}y = B^{-1}y$.*

PROOF. If $A \preceq B$ then for every $c \in \mathbb{R}^m$, $(c, Ac) \leq (c, Bc)$ and it follows that $\frac{1}{\gamma(A)} \leq \frac{1}{\gamma(B)}$. Clearly $A^{-1}y = B^{-1}y$ implies that $\gamma(A) = \gamma(B)$. On the other hand if $\gamma(A) = \gamma(B)$, the inequalities $\frac{1}{\gamma(A)} \leq (c(B), Ac(B)) \leq (c(B), Bc(B)) = \frac{1}{\gamma(B)}$ imply that $c(A) = c(B)$ and the result follows. \square

Lemma 17 *The function γ is convex and the function γ^{-1} concave. Moreover, for every $A, B \in \mathcal{L}_+(\mathbb{R}^m)$, $\lambda \in [0, 1]$, we have that*

$$\frac{1}{\gamma(\lambda A + (1-\lambda)B)} = \lambda \frac{1}{\gamma(A)} + (1-\lambda) \frac{1}{\gamma(B)} \quad (18)$$

if and only if $c(A) = c(B) = c(\lambda A + (1-\lambda)B)$.

PROOF. For every $\lambda \in [0, 1]$ we define the matrix $D_\lambda = \lambda A + (1 - \lambda)B$ and for all $c \in \mathbb{R}^m$ for which $(c, y) = 1$ note that

$$(c, D_\lambda c) = \lambda(c, Ac) + (1 - \lambda)(c, Bc) \geq \lambda \frac{1}{\gamma(A)} + (1 - \lambda) \frac{1}{\gamma(B)}. \quad (19)$$

Consequently, we have that $\frac{1}{\gamma(D_\lambda)} \geq \lambda \frac{1}{\gamma(A)} + (1 - \lambda) \frac{1}{\gamma(B)}$, showing that γ^{-1} is concave. Alternatively, equation (14) expresses $\gamma^{-1}(A)$ as the minimum of a family of functions which are linear in the matrix A and hence γ^{-1} is concave. Similarly, using this equation we have that

$$\gamma(A) = \max \{ (c, Ac)^{-1} : c \in \mathbb{R}^m, (c, y) = 1 \}$$

thereby expressing γ as a maximum of a family of convex functions.

If (18) holds, we choose $c = c_\lambda := c(D_\lambda)$ in (19) and conclude by the uniqueness of the vector $c(A)$ in equation (15) that $c_\lambda = c(A) = c(B)$. Conversely, when this conclusion holds we have that

$$\begin{aligned} \frac{1}{\gamma(D_\lambda)} &= \lambda(c_\lambda, Ac_\lambda) + (1 - \lambda)(c_\lambda, Bc_\lambda) \\ &= \lambda(c(A), Ac(A)) + (1 - \lambda)(c(B), Bc(B)) \\ &= \lambda \frac{1}{\gamma(A)} + (1 - \lambda) \frac{1}{\gamma(B)} \end{aligned}$$

which concludes the proof. \square

Lemma 16 and 17 established that the function $\phi : \mathcal{L}_+(\mathbb{R}^m) \rightarrow \mathbb{R}$ defined, for some $d \in \mathbb{R}^m$ and all $A \in \mathcal{L}_+(\mathbb{R}^m)$, as $\phi(A) = (d, A^{-1}d)$ is non-increasing and convex (see also the work of Marshall and Olkin, 1979).

Proposition 15 and Lemma 14 connects minimal norm interpolation to square loss regularization. This connection allows us in this section to turn our attention to the function $\rho : \mathcal{A}(X) \rightarrow \mathbb{R}_+$ and consider the variational problem

$$\rho(\mathcal{K}) := \inf \{ \rho(K) : K \in \mathcal{K} \} \quad (20)$$

where \mathcal{K} is a prescribed set of kernels. The approach of Lemma 2 applies directly to establish the following lemma.

Lemma 18 *If \mathcal{K} is a compact and convex set of kernels in $\mathcal{A}_+(X)$ then the minimum of (20) exists.*

Our next result describes the solution of the problem of determining $\rho(\mathcal{K})$ for the case that $\mathcal{K} = \text{co}\mathcal{K}_u$ where $\mathcal{K}_u = \{K_\ell : \ell \in \mathbb{N}_n\}$ is a prescribed finite subset of $\mathcal{A}_+(X)$. In its presentation we use the notion $K_{\mathbf{x}, \ell}$ for the matrix $(K_\ell)_{\mathbf{x}}$.

Theorem 19 *If $\mathcal{K}_u = \{K_j : j \in \mathbb{N}_n\} \subset \mathcal{A}_+(X)$ there exists a kernel $\hat{K} = \sum_{j \in J} \lambda_j K_j \in \text{co}\mathcal{K}_u$, where $J \subseteq \mathbb{N}_n$, $\text{card}(J) \leq \min(m + 1, n)$ with $\sum_{j \in J} \lambda_j = 1$ such that, for every $j \in J$, $\lambda_j > 0$,*

$$(\hat{c}, K_{\mathbf{x}, j} \hat{c}) = \max \{ (\hat{c}, K_{\mathbf{x}, \ell} \hat{c}) : \ell \in \mathbb{N}_n \}, \quad \hat{c} = c(\hat{K}_{\mathbf{x}}),$$

$$\rho(\mathcal{K}) = \rho(\hat{K}) = (y, \hat{K}_{\mathbf{x}}^{-1} y)$$

and for every $c \in \mathbb{R}^m$ with $(c, y) = 1$ and every $K \in \text{co}\mathcal{K}_u$

$$(\hat{c}, K_{\mathbf{x}}\hat{c}) \leq (\hat{c}, \hat{K}_{\mathbf{x}}\hat{c}) \leq (c, \hat{K}_{\mathbf{x}}c). \quad (21)$$

Inequality (21) expresses the fact that the pair (\hat{c}, \hat{K}) is a *saddle point* for the minimax problem

$$\tilde{\rho}^{-1} = \min \{ \max \{ (c, K_{\mathbf{x}}c) : K \in \text{co}\mathcal{K}_u \} : c \in \mathbb{R}^m, (c, y) = 1 \}.$$

The existence of (\hat{c}, \hat{K}) above implies that the minimum and maximum can be interchanged, that is,

$$\max \{ \min \{ (c, K_{\mathbf{x}}c) : c \in \mathbb{R}^m, (c, y) = 1 \} : K \in \text{co}\mathcal{K}_u \} \quad (22)$$

$$= \min \{ \max \{ (c, K_{\mathbf{x}}c) : K \in \text{co}\mathcal{K}_u \} : c \in \mathbb{R}^m, (c, y) = 1 \}. \quad (23)$$

Moreover, any \hat{c} and \hat{K} with the properties described in Theorem 19 is a saddle point of this minimax problem. Indeed, the upper bound in (21) follows from the definition of the vector \hat{c} and the function γ defined earlier, see equations (14) and (15). The lower bound follows from the fact that for any $K \in \text{co}\mathcal{K}_u$ we have that $(\hat{c}, K_{\mathbf{x}}\hat{c}) \leq \max \{ (\hat{c}, K_{\mathbf{x},\ell}\hat{c}) : \ell \in \mathbb{N}_n \}$.

Let us now turn to the existence of \hat{K} . Note that by equation (14) and Proposition 15 the expression in (22) is $1/\rho(\mathcal{K})$, the reciprocal of the quantity of interest to us. It is the quantity in equation (23) which we examine in the proof of Theorem 19 and it has been denoted by $\tilde{\rho}^{-1}$. A consequence of Theorem 19 is that $\tilde{\rho} = \rho(\mathcal{K})$. Certainly, by their definitions it is clear that $\tilde{\rho} \leq \rho(\mathcal{K})$.

We now present the proof of Theorem 19.

PROOF. Let \tilde{c} be a solution to problem (23). We define the set

$$J^* \equiv J(\tilde{c}) := \{ j : j \in \mathbb{N}_n, (\tilde{c}, K_{\mathbf{x},j}\tilde{c}) = \max \{ (\tilde{c}, K_{\mathbf{x},i}\tilde{c}) : i \in \mathbb{N}_n \} \}$$

the convex function $\varphi : \mathbb{R}^m \rightarrow \mathbb{R}$ by setting for each $c \in \mathbb{R}^m$, $\varphi(c) := \max \{ (c, K_{\mathbf{x},j}c) : j \in \mathbb{N}_n \}$ and note that by Lemma 24 the directional derivative of φ along the “direction” $d \in \mathbb{R}^m$, denoted by $\varphi'_+(c; d)$, is given by

$$\varphi'_+(c; d) = 2 \max \{ (d, K_{\mathbf{x},j}c) : j \in J(c) \}.$$

Since \tilde{c} is a minimum for (14) we have that

$$\max \{ (d, K_{\mathbf{x},j}\tilde{c}) : j \in J^* \} \geq 0$$

for every $d \in \mathbb{R}^m$ such that $(d, y) = 0$. Let \mathcal{M} be the convex hull of the set of vectors $\mathcal{N} := \{ K_{\mathbf{x},j}\tilde{c} : j \in J^* \} \subset \mathbb{R}^m$. Since $\mathcal{M} \subseteq \mathbb{R}^m$, by the Caratheodory theorem (see, for example, Rockafellar, 1970, Ch. 17) every vector in \mathcal{M} can be expressed as a convex combination of at most $q := \min(m+1, |J^*|) \leq \min(m+1, n)$ elements of \mathcal{N} . We will show that \mathcal{M} intersects the line spanned by the vector y . Indeed, if these two sets did not intersect then there exists a hyperplane $\{ c : c \in \mathbb{R}^m, (w, c) + \alpha = 0 \}$, where $\alpha \in \mathbb{R}$, $w \in \mathbb{R}^m$, which strictly separates them, that is,

$$(w, ty) + \alpha > 0, \quad t \in \mathbb{R}$$

and

$$(w, K_{\mathbf{x},j}\tilde{c}) + \alpha < 0, \quad j \in J^*, \quad (24)$$

(see, for example, Royden, 1988).

The first condition, for $t = 0$, implies that $\alpha > 0$ and since t can take any real value we also have that $(w, y) = 0$. Consequently, from equation (24) we get that

$$\max\{(w, K_{\mathbf{x},j}\tilde{c}) : j \in J^*\} < 0$$

which contradicts our hypothesis that \tilde{c} is a minimum. Thus, it must be the case that $t_0y \in \mathcal{M}$ for some $t_0 \in \mathbb{R}$, that is,

$$t_0y = \sum_{j \in J} \gamma_j K_{\mathbf{x},j} \tilde{c} \tag{25}$$

for some subset J of J^* of cardinality at most q and positive constants γ_j with $\sum_{j \in J} \gamma_j = 1$. Taking the inner product of both sides of equation (25) with \tilde{c} , and recalling the fact that $(\tilde{c}, y) = 1$ we obtain that $t_0 = \tilde{\rho}^{-1}$. Setting

$$\hat{K} := \sum_{j \in J} \gamma_j K_j$$

we have from (25) that $\tilde{c} = \tilde{\rho}^{-1} \hat{K}_{\mathbf{x}}^{-1} y$, and $\tilde{\rho} = (y, \hat{K}_{\mathbf{x}}^{-1} y)$. Therefore, by Proposition 15 we conclude that $\tilde{\rho} = \rho(\hat{K})$ and $\tilde{c} = \hat{c}$ where \hat{c} is defined in the theorem. In particular, we obtain $\tilde{\rho} \geq \rho(\mathcal{K})$ and so by our previous remarks just before the beginning of the proof, we conclude that $\tilde{\rho} = \rho(\mathcal{K})$. \square

Recall, that earlier we introduced the class $\mathcal{K}(G)$ induced by a continuous mapping $G : \Omega \rightarrow \mathcal{A}_+(\mathcal{X})$ where Ω is a compact Hausdorff space. Theorem 15 extends to this generality. No essential difference occur in the proof. However, the conclusion is striking. Not only do we characterize the optimal kernel $\hat{K} \in \mathcal{K}(G)$ but we show that it comes from a *discrete* probability measure $\hat{p} \in \mathcal{M}(\Omega)$ with *at most* $m + 1$ atoms, that is, $\hat{K} = \int_{\Omega} G(\omega) d\hat{p}(\omega)$.

Theorem 20 *If Ω is a compact Hausdorff topological space and $G : \Omega \rightarrow \mathcal{A}_+(\mathcal{X})$ is continuous then there exists a kernel $\hat{K} = \int_{\Omega} G(\omega) d\hat{p}(\omega) \in \mathcal{K}(G)$ such that \hat{p} is a discrete probability measure in $\mathcal{M}(\Omega)$ with at most $m + 1$ atoms. Moreover, for any atom $\hat{\omega} \in \Omega$ of \hat{p} , we have that*

$$(\hat{c}, G_{\mathbf{x}}(\hat{\omega})\hat{c}) = \max\{(\hat{c}, G_{\mathbf{x}}(\omega)\hat{c}) : \omega \in \Omega\}, \quad \hat{c} = c(\hat{K}_{\mathbf{x}}),$$

$$\rho(\mathcal{K}) = \rho(\hat{K}) = (y, \hat{K}_{\mathbf{x}}^{-1} y)$$

and for every $c \in \mathbb{R}^m$ with $(c, y) = 1$ and every $K \in \mathcal{K}(G)$

$$(\hat{c}, K_{\mathbf{x}}\hat{c}) \leq (\hat{c}, \hat{K}_{\mathbf{x}}\hat{c}) \leq (c, \hat{K}_{\mathbf{x}}c).$$

PROOF. Let \tilde{c} be a solution to problem (23) where $co\mathcal{K}_{\mathbf{v}}$ is replaced by $\mathcal{K}(G)$ and define the set

$$\Omega^* \equiv \Omega(\tilde{c}) := \{\tau : \tau \in \Omega, (\tilde{c}, G_{\mathbf{x}}(\tau)\tilde{c}) = \max\{(\tilde{c}, G_{\mathbf{x}}(\omega)\tilde{c}) : \omega \in \Omega\}\}.$$

where we denoted the matrix $(G(\omega))_{\mathbf{x}}$ by $G_{\mathbf{x}}(\omega)$. We define the convex function $\varphi : \mathbb{R}^m \rightarrow \mathbb{R}$ by setting for each $c \in \mathbb{R}^m$, $\varphi(c) := \max\{(c, G_{\mathbf{x}}(\omega)c) : \omega \in \Omega\}$ and note that by Lemma 24 the directional derivative of φ along the “direction” $d \in \mathbb{R}^m$, denoted by $\varphi'_+(c; d)$, is given by

$$\varphi'_+(c; d) = 2 \max\{(d, G_{\mathbf{x}}(\omega)c) : \omega \in \Omega^*\}.$$

Since \tilde{c} is a minimum for (14) we have that

$$\max\{(d, G_{\mathbf{x}}(\omega)\tilde{c}) : \omega \in \Omega(c)\} \geq 0$$

for every $d \in \mathbb{R}^m$ such that $(d, y) = 0$. Let \mathcal{M} be the convex hull of the set of vectors $\mathcal{N} := \{G_{\mathbf{x}}(\omega)\tilde{c} : \omega \in \Omega^*\} \subset \mathbb{R}^m$. Since $\mathcal{M} \subseteq \mathbb{R}^m$, by the Caratheodory theorem every vector in \mathcal{M} can be expressed as a convex combination of at most $m + 1$ elements of \mathcal{N} . We will show that \mathcal{M} intersects the line spanned by the vector y . Indeed, if these two sets did not intersect then there exist a hyperplane $(w, c) + \alpha = 0$, $\alpha \in \mathbb{R}$, $w \in \mathbb{R}^m$, which strictly separates them, that is,

$$(w, ty) + \alpha > 0, \quad t \in \mathbb{R}$$

and

$$(w, G_{\mathbf{x}}(\omega)\tilde{c}) + \alpha < 0, \quad \omega \in \Omega^*, \quad (26)$$

(see Royden, 1988).

The first condition, for $t = 0$, implies that $\alpha > 0$ and since t can take any real value we also have that $(w, y) = 0$. Consequently, from equation (26) we get that

$$\max\{(w, G_{\mathbf{x}}(\omega)\tilde{c}) : \omega \in \Omega^*\} < 0.$$

which contradicts our hypothesis that \tilde{c} is a minimum. Thus, it must be the case that $t_0 y \in \mathcal{M}$ for some $t_0 \in \mathbb{R}$, that is,

$$t_0 y = \int_{\Omega} G_{\mathbf{x}}(\omega)\tilde{c} d\hat{p}(\omega) \quad (27)$$

where $\hat{p} \in \mathcal{M}(\Omega)$ is a discrete probability measure with at most $m + 1$ atoms. Taking the inner product of both sides of equation (27) with \tilde{c} , and recalling the fact that $(\tilde{c}, y) = 1$ we obtain that $t_0 = \tilde{\rho}^{-1}$. Setting

$$\hat{K} := \int_{\Omega} G_{\mathbf{x}}(\omega) d\hat{p}(\omega)$$

we have from (27) that $\tilde{c} = \tilde{\rho}^{-1}\hat{K}^{-1}y$, and $\tilde{\rho} = (y, \hat{K}^{-1}y)$. Therefore, by Proposition 15 we conclude that $\tilde{\rho} = \rho(\hat{K})$ and $\tilde{c} = \hat{c}$ where \hat{c} is defined in the theorem. In particular, we obtain $\tilde{\rho} \geq \rho(\mathcal{K})$ and so by our previous remarks we conclude that $\tilde{\rho}_0 = \rho(\mathcal{K})$. \square

This theorem applies to the Gaussian kernel.

Corollary 21 *If $a > 0$ and $N : [a, b] \rightarrow \mathcal{A}_+(\mathcal{X})$ is defined as*

$$N(\omega)(x, t) = e^{-\omega\|x-t\|^2}, \quad x, t \in \mathbb{R}^d, \quad \omega \in \mathbb{R}_+$$

then there exists a kernel $\hat{K} = \int_{\Omega} N(\omega) d\hat{p}(\omega) \in \mathcal{K}(N)$ such that \hat{p} is a discrete probability measure in $\mathcal{M}(\Omega)$ with at most $m + 1$ atoms. Moreover, for any atom $\hat{\omega} \in \Omega$ of \hat{p} , we have that

$$(\hat{c}, N_{\mathbf{x}}(\hat{\omega})\hat{c}) = \max\{(\hat{c}, N_{\mathbf{x}}(\omega)\hat{c}) : \omega \in \Omega\}, \quad \hat{c} = c(\hat{K}_{\mathbf{x}}),$$

$$\rho(\mathcal{K}(N)) = \rho(\hat{K}) = (y, \hat{K}_{\mathbf{x}}^{-1}y)$$

and for every $c \in \mathbb{R}^m$ and $K \in \mathcal{K}(N)$ we have that

$$(\hat{c}, K_{\mathbf{x}}\hat{c}) \leq (\hat{c}, \hat{K}_{\mathbf{x}}\hat{c}) \leq (c, \hat{K}_{\mathbf{x}}c).$$

We note that, in view of equations (13) and (17), Theorem 19 and Theorem 20 apply directly, up to an unimportant constant μ , to the square loss functional by merely adding the kernel $\mu\Delta$ to the class of kernels considered in these theorems. That is, we minimize the quantity in equation (17) over the compact convex set of kernels

$$\tilde{\mathcal{K}} = \{\tilde{K} : \tilde{K} = K + \mu\Delta, K \in \mathcal{K}\}$$

where the kernel Δ is defined in equation (8).

An important example of the above construction is to choose K_j to be polynomials on \mathbb{R}^d , namely $K_j(x, t) = (x, t)^j, x, t \in \mathbb{R}^d$. From a practical point of view we should limit the range of the index j and therefore Theorem 19 adequately covers this case. On the contrary if we decide to use, as it is done often, Gaussians, there arises not only how many Gaussians to choose but also which ones to choose. This raises the question of looking at the *whole class of radial basis functions* and trying to choose the best kernel amongst this class. To this end, we recall a beautiful result of Schoenberg (1938). Let φ be a real-valued function defined on \mathbb{R}_+ which we normalize so that $\varphi(0) = 1$. We form a kernel K on \mathbb{R}^d by setting for each $x, t \in \mathbb{R}^d$ $K(x, t) = \varphi(\|x - t\|^2)$. Schoenberg showed that K is positive definite for *any* d if and only if there is a probability measure p on \mathbb{R}_+ such that

$$K(x, t) = \int_{\mathbb{R}_+} e^{-\sigma\|x-t\|^2} dp(\sigma), \quad x, t \in \mathbb{R}^d.$$

Note that the set \mathbb{R}_+ is *not* compact and the kernel $N(0)$ is not in $\mathcal{A}_+(\mathbb{R}^d)$. Therefore, on both accounts Theorem 20 does not apply in this circumstance unless, of course, we impose a positive lower bound and a finite upper bound on the variance of the Gaussian kernels $N(\omega)$. We may overcome this difficulty by a limiting process which can handle kernel maps on *locally compact Hausdorff spaces*. This will lead us to an extension of Theorem 20 where Ω is locally compact. However, we only describe our approach in detail for the Gaussian case and $\Omega = \mathbb{R}_+$. An important ingredient in this discussion presented below is that $N(\infty) = \Delta$.

For every $\ell \in \mathbb{N}$ we consider the Gaussian kernel map on the interval $\Omega_\ell := [\ell^{-1}, \ell]$ and appeal to Theorem 20 to produce a sequence of kernels $\hat{K}_\ell = \int_{\Omega_\ell} N(\omega) dp_\ell(\omega)$ with the properties described there. In particular, p_ℓ is a discrete probability measure with at most $m + 1$ atoms, a number *independent* of ℓ . Let us examine that may happen as ℓ tends towards infinity. Each of the atoms of p_ℓ as well their corresponding weights have subsequences which converge. Some atoms may converge to zero while others to infinity. In either case, the Gaussian kernel map *approaches a limit*. Therefore, we can extract a convergent subsequence $\{p_{n_\ell} : \ell \in \mathbb{N}\}$ of probability measures and kernels $\{K_{n_\ell} : \ell \in \mathbb{N}\}$ such that $\lim_{\ell \rightarrow \infty} p_{n_\ell} = \hat{p}$, $\lim_{\ell \rightarrow \infty} K_{n_\ell} = \hat{K}$, and $\hat{K} = \int_{\mathbb{R}_+} N(\omega) \hat{p}(\omega)$ with the provision that \hat{p} may have atoms at either zero or infinity. In either case, we replace the Gaussian by its limit, namely $N(0)$, the identically one kernel, or $N(\infty)$, the delta kernel, in the integral which defines \hat{K} . *All* of the properties described in Theorem 20 and remarks following it hold for \hat{K} because of the simplicity of the objective function for the minimax problem studied there. Hence \hat{K} is the *best radial kernel*.

4. Discussion

In this final section we comment on two recent papers related to ours, present some numerical simulations and outline possible extensions of the ideas presented above.

4.1 Related Works

Lanckriet et al. (2004) address learning kernels in the context of transductive learning, that is, learning the value of a function at a finite set of test points. In this case the kernel is computed only on the training and test sets and, so, it is regarded as a matrix. The authors propose different criteria to find a positive semi-definite kernel matrix and discuss how these can be casted as positive semi-definite programming problems. For example, they maximize the *margin* of a binary support vector machine (SVM) trained with the kernel K , which is the square root of the reciprocal of the quantity defined by the equation

$$\rho_{hard}(K) = \min \{ \|f\|_K^2 : y_j f(x_j) \geq 1, j \in \mathbf{N}_m \}. \quad (28)$$

where $y_j \in \{-1, 1\}$ are class labels, (see, for example, Vapnik, 1998). The margin is the maximum distance of the closed point, relative to a set of labeled points, amongst all separating functions in the RKHS. These functions are hyperplanes in the space spanned by the features associated to a Mercer expansion of the kernel K . When the optimal separating hyperplane does not exist, the standard approach is to relax the separation constraints in problem (28) to obtain the so-called soft margin SVM,

$$\rho_{soft}(K) := \min \left\{ \sum_{j \in \mathbf{N}_m} \xi_j + \mu \|f\|_K^2 : y_j f(x_j) \geq 1 - \xi_j, \xi_j \geq 0, j \in \mathbf{N}_m, f \in \mathcal{H}_K \right\}. \quad (29)$$

These two problems are related. Indeed, if problem (28) admits a solution, that is, the constraints are feasible, problem (29) gives the same solution provided the parameter μ is small enough.

Lanckriet et al. (2004) consider the minimization problem (29) when \mathcal{K} is a set of positive semi-definite matrices which are linear combinations of some prescribed matrices $K_j, j \in \mathbf{N}_n$. In particular, if K_j are positive semi-definite \mathcal{K} could be the set of convex combination of such matrices. They show that $\rho_{soft}(K)$ is convex in $K \in \mathcal{K}$. Our observations in Section 2 confirm that the margin and the soft margin are convex functions of the kernel. Indeed, problem (29) is equivalent to the variational problem (1) when Q is the *hinge error function* defined on \mathbf{R}^m by

$$Q(w) := \sum_{j \in \mathbf{N}_n} (1 - y_j w_j)_+, \quad w := (w_j : j \in \mathbf{N}_m)$$

where $t_+ := \max(0, t), t \in \mathbf{R}$, (see, for example, Evgeniou, Pontil and Poggio, 2000).

Ong, Smola and Williamson (2003) consider learning a kernel function rather than a kernel matrix. They choose a set \mathcal{K} in the space of kernels which are in a Hilbert space of functions generated by a so-called hyper-kernel. This is a kernel $H : \mathcal{X}^2 \times \mathcal{X}^2 \rightarrow \mathbf{R}$, where $\mathcal{X}^2 = \mathcal{X} \times \mathcal{X}$, with the property that, for every $(x, t) \in \mathcal{X}^2, H((x, t), (\cdot, \cdot))$ is a kernel on $\mathcal{X} \times \mathcal{X}$. This construction includes convex combinations of a possibly infinite number of kernels provided they are *pointwise nonnegative*. For example Gaussian kernels or polynomial kernels with even degree satisfy this assumption although those with odd degree, such as linear kernels or other radial kernels do not.

4.2 Numerical Simulations

In this section we discuss two numerical simulations we carried out to compute a convex combination of a finite set of kernels $\{K_\ell : \ell \in \mathbf{N}_n\}$ which minimizes the square loss regularization functional

μ	10^{-4}	10^{-3}	10^{-2}	0.1	1	10
Method 1	2.41 (1.04)	1.69 (0.68)	0.60 (0.11)	0.27 (0.08)	0.26 (0.05)	3.20 (0.48)
Method 2	1.54 (0.58)	0.91 (0.22)	0.47 (0.08)	0.40 (0.07)	0.61 (0.11)	3.80 (0.58)
Method 3	4.65 (7.81)	0.95 (1.24)	0.21 (0.06)	0.10 (0.05)	0.12 (0.08)	2.40 (0.60)

Table 1: *Experiment 1: Average mean square error with its standard deviation (in parenthesis) for methods 1 to 3 for different values of the regularization parameter μ (see text for the description). The unit measure for the errors is 10^{-3} .*

S_μ in equation (5). For this purpose, we use an interior point method, that is, we define, for every $\lambda = (\lambda_\ell : \ell \in \mathbb{N}_n) \in \mathbb{R}^n$, the penalized function

$$F_\nu(\lambda) := S_\mu \left(\sum_{\ell \in \mathbb{N}_n} \lambda_\ell K_\ell \right) - \nu \sum_{\ell \in \mathbb{N}_n} \ln \lambda_\ell \quad (30)$$

where ν is a positive parameter and solve the variational problem

$$\min \left\{ F_\nu(\lambda) : \lambda \in \mathbb{R}^n, \sum_{\ell \in \mathbb{N}_n} \lambda_\ell = 1 \right\}. \quad (31)$$

Clearly, when ν is small the solution to this problem is close to a minimizer of S_μ , although the penalty term in (30) forces this solution to be interior to the set $\{\lambda : \sum_{\ell \in \mathbb{N}_n} \lambda_\ell = 1, \lambda_\ell \geq 0, \ell \in \mathbb{N}_n\}$. In order to reach such a minimizer we choose an iteration number $R \in \mathbb{N}$ and iteratively compute the solution to problem (31) for a decreasing sequence of values of the parameter ν . Specifically we set, for $r \in \mathbb{N}_R$, $\nu_r = \bar{\nu} A^{r-1}$ where $\bar{\nu}$ is the initial value of ν and $A \in (0, 1)$ is some prescribed parameter. The optimality conditions for problem (31) (see, for example, Rockafellar, 1970; Borwein and Lewis, 2000) are given by the system of *non-linear* equations

$$\begin{aligned} \nabla F_\nu - \eta e &= 0 \\ -(e, \lambda) + 1 &= 0 \end{aligned}$$

where e is the vector in \mathbb{R}^n all of whose components are one and $\eta \in \mathbb{R}$ is the Lagrange multiplier associated to the equality constraint in that problem. We solve these equations by a Newton method (see, for example, Mangasarian, 1994) which consists in iteratively solving the system of *linear* equations

$$\begin{aligned} \nabla^2 F_\nu(\hat{\lambda}) \Delta_\lambda - \Delta_\eta e &= \hat{\eta} e - \nabla F_\nu(\hat{\lambda}) \\ -(e, \Delta_\lambda) &= 0 \end{aligned}$$

to obtain the vector $\Delta_\lambda \in \mathbb{R}^n$ and $\Delta_\eta \in \mathbb{R}$, where $\hat{\lambda}$ and $\hat{\eta}$ are the previous values of λ and η . We then update the parameters as $\lambda = \hat{\lambda} + \alpha \Delta_\lambda$ and $\eta = \hat{\eta} + \alpha \Delta_\eta$, where, in order to insure that $\lambda \in [0, 1]^n$, we have set $\alpha := \min(1, 0.5 \max\{\alpha > 0 : \hat{\lambda} + \alpha \Delta_\lambda \in [0, 1]^n\})$. In our experiments below we choose $R = 5$, $\bar{\nu} = 10$ and $A = 0.5$.

In both experiments we tried to learn a target function $f : [0, 2\pi] \rightarrow \mathbb{R}$ from a set of its samples. In the first experiment we fixed $f(x) = \frac{1}{10}(x + 2(e^{-8(\frac{4}{3}\pi-x)^2} - e^{-8(\frac{\pi}{2}-x)^2} - e^{-8(\frac{3}{2}\pi-x)^2}))$, $x \in [0, 2\pi]$, and,

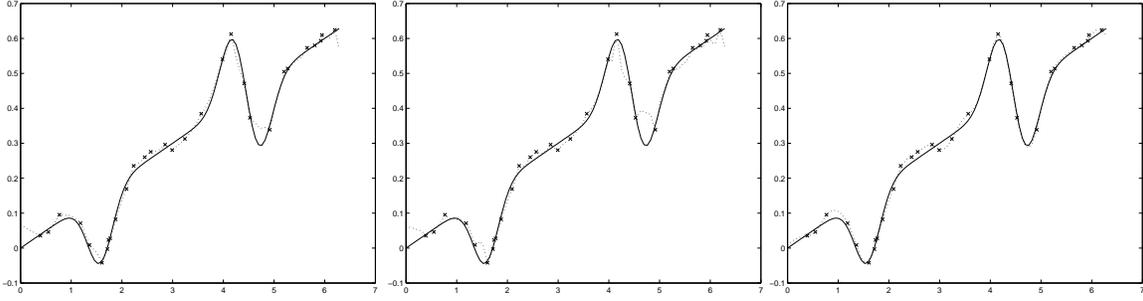


Figure 1: Experiment 1: function learned by method 1 (left), method 2 (center) and method 3 (right). Regularization parameter is $\mu = 0.1$, the number of training points is 50. Solid line is the target function, crosses are the sampled points and the dotted line is the method used. The vertical scale has been reduce

μ	10^{-4}	10^{-3}	10^{-2}	0.1	1	10
Method 1	3.46 (1.39)	3.46 (1.39)	3.45 (1.38)	3.35 (1.35)	2.64 (1.10)	14.1 (10.3)
Method 2	4.46 (1.82)	4.46 (1.79)	3.85 (1.18)	3.78 (1.03)	4.00 (1.02)	62.6 (5.11)
Method 3	0.52 (0.56)	0.51 (0.56)	0.51 (0.55)	0.51 (0.57)	0.53 (0.63)	3.51 (1.47)

Table 2: Experiment 2: Average mean square error with its standard deviation (in parenthesis) for methods 1 to 3 for different values of the regularization parameter μ (see text for the description). The unit measure for the errors is 10^{-3} .

for every $x, t \in [0, 2\pi]$, we set $K_\ell(x, t) = (xt)^{\ell-1}$ if $\ell \in \{1, 2, 3\}$ and $K_\ell(x, t) = e^{-\omega_\ell(x-t)^2}$ if $\ell \in \{4, 5, 6\}$ where $\omega_\ell = 2^{8-5(\ell-4)}$. We generated a training set of fifty points $\{(x_j, y_j) : j \in \mathbf{N}_{50}\} \subset [0, 2\pi] \times \mathbf{R}$ obtained by sampling f with noise. Specifically, we choose x_j uniformly distributed in the interval $[0, 2\pi]$ and $y_j = f(x_j) + \varepsilon$ with ε also uniformly sampled in the interval $[-0.02, 0.02]$. We then computed on a test set of 100 samples the mean square error between the target function f and the function learned from the training set for different values of the parameter μ . We compare three methods. *Method 1* is our proposed approach, *method 2* is the average of the kernels, that is we use the kernel $K = \frac{1}{n} \sum K_\ell$ and *method 3* is the kernel $K = K_2 + K_5$, the “ideal” kernel, that is, the kernel used to generate the target function. The results are shown in Table 1. Figure 1 shows the function learned by each method.

In our second experiment we fixed $f(x) = \sin(x) + \frac{1}{2}\sin(3x)$, $x \in [0, 2\pi]$ and $K_\ell(x, t) = \sin(\ell x) \sin(\ell t)$, $x, t \in [0, 2\pi]$, $\ell \in \mathbf{N}_n$. The set up is similar to that in Experiment 1. *Method 1* is our proposed approach, *method 2* is the average of the kernels and *method 3* is the ideal kernel given by $K(x, t) = \frac{2}{3} \sin(x) \sin(t) + \frac{1}{3} \sin(3x) \sin(3t)$. The noise ε is now uniformly sampled in the interval $[-0.2, 0.2]$. The results are reported in Table 2. Figure 2 shows the function learned by each method.

4.3 Extensions

We discuss some extensions of the problems studied in this paper. The first one that comes to mind is obtained by taking the expectation of the functional (4) with respect to a probability measure P on \mathbf{R}^m , that is,

$$Q_\mu^{av}(K) := \int_{\mathbf{R}^m} Q_\mu(K, y) P(y) dy, \quad K \in \mathcal{K} \tag{32}$$

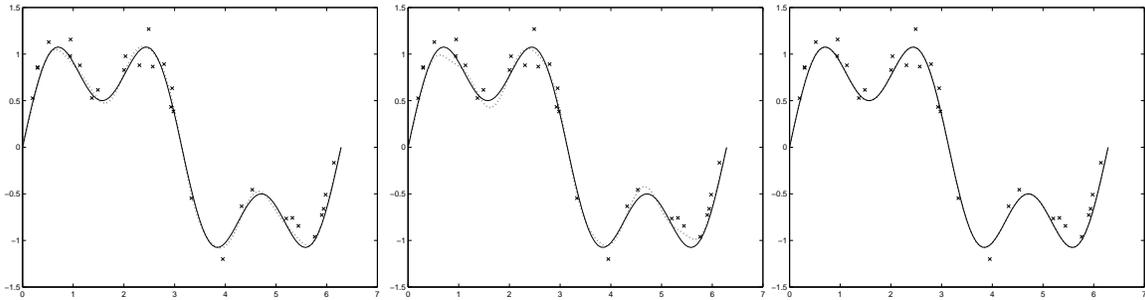


Figure 2: *Experiment 2: function learned by method 1 (left), method 2 (center) and method 3 (right). Regularization parameter is $\mu = 0.1$, the number of training points is 50. Solid line is the target function, crosses are the sampled points and the dotted line is the method used.*

where we indicated the dependency of $Q_\mu(K)$ on y by writing $Q_\mu(K, y)$. Since $Q_\mu(K, y)$ is convex in K for each $y \in \mathbb{R}^m$ so is $Q_\mu^{av}(K)$. We then minimize $Q_\mu^{av}(K)$ over $K \in \mathcal{K}$. For the square loss regularization we obtain that

$$S_\mu^{av}(K) = \mu \text{trace}((K_{\mathbf{x}} + \mu I)^{-1} \Sigma) \quad (33)$$

where Σ is the correlation matrix of P . Minimizing the quantity (33) over a convex class \mathcal{K} may be valuable in image reconstruction and compression where we are provided with a collection of images and we wish to find a good average representation for them. In this case the input $\mathbf{x} = \{x_i : i \in \mathbb{N}_m\}$ represents the locations of the image pixels. For gray level images we can assume that $y \in [0, 1]^m$ and therefore we should choose P to have support on $[0, 1]^m$. Thus, if $\{y^\ell : \ell \in \mathbb{N}_n\}$ is a sample of such images with $n < m$ and Σ is the rank n empirical correlation matrix our goal is to find a kernel which well-represents this collection on the average.

Another approach is provided by replacing the average in equation (32) with the maximum over all y with bounded norm, that is, we minimize the functional

$$Q_\mu^{max}(K) := \max\{Q_\mu(K, y) : \|y\| \leq 1\}, \quad K \in \mathcal{K}.$$

Again, this function is convex in K . In particular, for square loss regularization and the Euclidean norm on \mathbb{R}^m we obtain

$$\max\{S_\mu(K, y) : \|y\| \leq 1\} = \max\{\mu(y, (K_{\mathbf{x}} + \mu I)^{-1} y) : \|y\| \leq 1\} = \frac{\mu}{\lambda_{min}(K_{\mathbf{x}}) + \mu}$$

where $\lambda_{min}(K_{\mathbf{x}})$ is the smallest eigenvalue of the matrix $K_{\mathbf{x}}$. Consequently, we have that

$$\min\{\max\{S_\mu(K, y) : \|y\| \leq 1\} : K \in \mathcal{K}\} = \frac{\mu}{\max\{\lambda_{min}(K_{\mathbf{x}}) : K \in \mathcal{K}\} + \mu}.$$

It is well-known that $\lambda_{min}(K_{\mathbf{x}})$ is a concave function of $K_{\mathbf{x}}$, (see, for example, Marshall and Olkin, 1979, p. 475). Therefore, our results provide an alternate proof of this fact.

We also remark that instead of learning a function f from function values the information operator I can be of the form $I(f) = ((g_j, f) : j \in \mathbb{N}_m)$, $f \in \mathcal{H}$, where $\{g_j : j \in \mathbb{N}_m\}$ is a set of prescribed functions in a Hilbert space, see the work of Micchelli and Pontil (2004) for a discussion. In this

case, the matrix $K_{\mathbf{x}}$ becomes the Gram matrix of these functions. The previous sections considered the choice $g_j = K(x_j, \cdot)$ and the Gram matrix is $K_{\mathbf{x}}$. This extension has wide applications in inverse problems, for example for computing the solution of first order integral equations.

Lemma 17 indicates that $Q_\mu : \mathcal{A}_+(\mathcal{X}) \rightarrow \mathbb{R}_+$ is, generally, not strictly convex. We may modify the functional Q_μ with a penalty term which depends on the kernel matrix $K_{\mathbf{x}}$ to enforce uniqueness of the optimal kernel in \mathcal{K} . Therefore, we consider the variational problem

$$\min\{Q_\mu(K) + R(K_{\mathbf{x}})\} \tag{34}$$

where R is a strictly convex function on $\mathcal{L}(\mathbb{R}^m)$. In this case, the method of proof of Theorem 7 shows that the optimal kernel can be found as a convex combination of at most $\frac{1}{2}m(m+1)$ kernels. For example, we may choose $R(A) = \text{trace}(A^2)$, $A \in \mathcal{L}(\mathbb{R}^m)$.

The variational problem (34) may be a preferred approach for choosing an optimal kernel. Indeed, if Q vanishes at some point in \mathbb{R}^m and there is a kernel $K \in \mathcal{K}$ such that for all $t > 0$, $tK \in \mathcal{K}$ then it follows that $Q_\mu(\mathcal{K}) = 0$. This fact follows since $\lim_{t \rightarrow \infty} Q_\mu(tK) = 0$, by elementary properties of the norm in \mathcal{H}_K . However, if the kernels in \mathcal{K} have the property that $\sup_{K \in \mathcal{G}} \sup_{x \in \mathcal{X}} K(x, x) < \infty$, that is, they are uniformly bounded, the above circumstance cannot occur. This observation suggests that our criterion may be free from overfitting. Preliminary experiments with Gaussian kernels confirm that overfitting does not occur (Argyriou, Micchelli and Pontil, 2005). We leave for a future occasion a detailed investigation of this important issue.

As a final comment, let us point out that a kernel map can also be parameterized by matrices. For example, to each $A \in \mathcal{L}(\mathbb{R}^d)$ we define the linear kernel $K_A(x, t) = (x, At)$, $x, t \in \mathbb{R}^d$ and so our results apply to any convex compact subset of $\mathcal{L}(\mathbb{R}^d)$ for this kernel map. Another example are Gaussians parameterized by covariances $\Sigma \in \mathcal{L}(\mathbb{R}^d)$, that is,

$$N(\Sigma)(x, t) = \frac{1}{\sqrt{\det(\Sigma)(2\pi)^d}} e^{-(x-t, \Sigma^{-1}(x-t))}, \quad x, t \in \mathbb{R}^d.$$

For compact convex sets of covariances our results say that Gaussian mixture models give optimal kernels.

5. Conclusion

The intent of this paper is to enlarge the theoretical understanding of the study of optimal kernels via the minimization of a regularization functional. Our analysis of this problem builds upon and extends the work of Lanckriet et al. (2004) and Lin and Zhang (2003). In contrast to the point of view of these papers, our setting applies to convex combinations of kernels parameterized by a compact set. Our analysis establishes that the regularization functional Q_μ is convex in K and that any optimizing kernel can be expressed as the convex combination of at most $m+2$ basic kernels. We have also provided a detailed characterization of the resulting minimax problem for square loss regularization. We have only marginally addressed at this stage implementation and algorithms for the search of optimal kernels. Since the proofs provided in Theorems 19 and 20 are constructive it should be possible to make use of them to derive practical algorithms for learning an optimal kernel such as a mixture of Gaussians, see (Argyriou, Micchelli and Pontil, 2005) for some recent results in this direction. Finally, an important direction which has not been explored in this paper is that of deriving error bounds, see (Micchelli et al., 2005) for some very recent progress on this.

Acknowledgments

We are grateful to Mark Herbster of University College London (UCL) for a remark which lead to Lemma 25, Raphael Hauser of Oxford University for suggesting a method to minimize the square loss regularization functional and many useful observations and to Andreas Argyriou of UCL for many useful comments. We also wish to thank Cheng Soon Ong of the Australian National University for discussions on his work which relates to ours, and are grateful to the referees for helping us clarify our presentation.

This work was partially supported by NSF Grant Number ITR-0312113, EPSRC Grant Number GR/T18707/01 and by the IST Programme of the European Community, under the PASCAL Network of Excellence IST-2002-506778.

Appendix A

The first result we record here is a useful version of the classical von Neumann minimax theorem.

Theorem 22 *Let $f : \mathcal{A} \times \mathcal{B} \rightarrow \mathbb{R}$ where \mathcal{A} is a compact convex subset of a Hausdorff topological vector space \mathcal{X} and \mathcal{B} is a convex subset of a vector space \mathcal{Y} . If the function $x \mapsto f(x, y)$ is convex and lower semi-continuous for every $y \in \mathcal{B}$ and $y \mapsto f(x, y)$ is concave for every $x \in \mathcal{A}$ then we have that*

$$\min\{\sup\{f(x, y) : y \in \mathcal{B}\} : x \in \mathcal{A}\} = \sup\{\inf\{f(x, y) : x \in \mathcal{A}\} : y \in \mathcal{B}\} \quad (35)$$

Theorem 23 *Let $f : \mathcal{A} \times \mathcal{B} \rightarrow \mathbb{R}$ where \mathcal{A} is a closed convex subset of a Hausdorff topological vector space \mathcal{X} and \mathcal{B} is a convex subset of a vector space \mathcal{Y} . If the function $x \mapsto f(x, y)$ is convex and lower semi-continuous for every $y \in \mathcal{B}$, $y \mapsto f(x, y)$ is concave for every $x \in \mathcal{A}$ and there exists a $y_0 \in \mathcal{B}$ such that for all $\lambda \in \mathbb{R}$ the set*

$$\{x : x \in \mathcal{A}, f(x, y_0) \leq \lambda\}$$

is compact then there is an $x_0 \in \mathcal{A}$ such that

$$\sup\{f(x_0, y) : y \in \mathcal{B}\} = \sup\{\inf\{f(x, y) : x \in \mathcal{A}\} : y \in \mathcal{B}\}$$

in particular, (35) holds

Theorem 22 is subsumed by Theorem 23 whose proof can be found in (Aubin, 1982, Ch. 7). The hypothesis of lower semi-continuity means, for all $\lambda \in \mathbb{R}$ and $y \in \mathcal{B}$, that the set $\{x : x \in \mathcal{A}, f(x, y) \leq \lambda\}$ is a closed subset of \mathcal{A} .

The next result concerns differentiation of a “max” function. The version we use comes from (Micchelli, 1969). Let \mathcal{X} be a topological vector space. If g is a continuous real-valued function on \mathcal{X} , we define its right derivative at $x \in \mathcal{X}$ in the direction $y \in \mathcal{X}$ by the formula

$$g'_+(x, y) = \lim_{\varepsilon \rightarrow 0^+} \frac{g(x + \varepsilon y) - g(x)}{\varepsilon}$$

whenever it exists.

Lemma 24 *Let \mathcal{T} a compact set and $G(t, x)$ a real-valued function on $\mathcal{T} \times X$ such that, for every $x \in X$ $G(\cdot, x)$ is continuous on \mathcal{T} and, for every $t \in \mathcal{T}$, $G(t, \cdot)$ is convex on X . We define the real-valued convex function g on X by the formula*

$$g(x) := \max\{G(t, x) : t \in \mathcal{T}\}, \quad x \in X$$

and the set

$$M(x) := \{t : t \in \mathcal{T}, G(t, x) = g(x)\}.$$

Then the right derivative of g in the direction $y \in X$ is given by

$$g'_+(x, y) = \max\{G'_+(t, x, y) : t \in M(x)\}$$

where $G'_+(t, x, y)$ is the right derivative of G with respect to its second argument in the direction y .

PROOF. We first observe, for every $t \in M(x)$ and $\lambda > 0$, that

$$\frac{g(x + \lambda y) - g(x)}{\lambda} \geq \frac{G(t, x + \lambda y) - G(t, x)}{\lambda}$$

which, letting $\lambda \rightarrow 0^+$, implies that $g'_+(x, y) \geq G'_+(t, x, y)$ and, so,

$$g'_+(x, y) \geq \sup\{G'_+(t, x, y) : t \in M(x)\}.$$

To prove the reverse inequality we use the fact that if f is convex on $[0, \infty)$ and $f(0) = 0$ then $f(\lambda)/\lambda$ is a nondecreasing function of $\lambda > 0$. In particular, this is true for the function of λ defined, for every $x, y \in X$, as

$$\frac{g(x + \lambda y) - g(x)}{\lambda}.$$

Consequently, we obtain, for every $\lambda > 0$ that

$$\frac{g(x + \lambda y) - g(x)}{\lambda} \geq g'_+(x, y).$$

Now, we define

$$h(\lambda, t) := \frac{G(t, x + \lambda y) - g(x)}{\lambda}, \quad \lambda > 0$$

and observe that, for each $t \in \mathcal{T}$, it is a nondecreasing function of λ because

$$h(\lambda, t) = \frac{G(t, x + \lambda y) - G(t, x)}{\lambda} - \frac{g(x) - G(t, x)}{\lambda}.$$

Therefore, the sets $A_\lambda := \{t \in \mathcal{T} : h(\lambda, t) \geq g'_+(x, y)\}$ are nonempty, closed and nested for $\lambda > 0$ and, so, the compactness of \mathcal{T} implies that there exists a $t_0 \in \bigcap_{\lambda > 0} A_\lambda$, that is,

$$G(t_0, x + \lambda y) \geq \lambda g'_+(x, y) + g(x), \quad \lambda > 0.$$

Thus, $t_0 \in M(x)$ and $g'_+(x, y) \leq G'_+(t_0, x, y)$. □

We now present the proof of Lemma 8 in an extended form. To this end, we let r be any positive number and let

$$co_r \mathcal{K}_u := \left\{ K : K = \sum_{j \in \mathbf{N}_n} \lambda_j K_j, \lambda_\ell \geq 0, \ell \in \mathbf{N}_n, \sum_{j \in \mathbf{N}_n} \lambda_j^r = 1 \right\}.$$

Note that $co_1 \mathcal{K}_u = co \mathcal{K}_u$ where $\mathcal{K}_u = \{K_j : j \in \mathbf{N}_n\}$.

Lemma 25 *If $\mathcal{K}_w = \{K_j : j \in \mathbf{N}_n\}$ is a family of kernels on $\mathcal{X} \times \mathcal{X}$ and $f \in \bigoplus_{j \in \mathbf{N}_n} \mathcal{H}_{K_j}$, and $s := \frac{2r}{r+1}$ then*

$$\inf\{\|f\|_K : K \in \text{co}_r \mathcal{K}_w\} = \min \left\{ \left(\sum_{j \in \mathbf{N}_n} \|f_j\|^s \right)^{\frac{1}{s}} : f = \sum_{j \in \mathbf{N}_n} f_j, f_\ell \in \mathcal{H}_{K_\ell}, \ell \in \mathbf{N}_n \right\}.$$

PROOF. The first step is to appeal to a result of Aronszajn, (see Aronszajn, 1950, p. 352-3), which states that for any $f \in \bigoplus_{j \in \mathbf{N}_n} \mathcal{H}_{K_j}$ we have for $K = \sum_{j \in \mathbf{N}_n} \lambda_j K_j$, with $\lambda_\ell > 0$, $\ell \in \mathbf{N}_n$ that

$$\|f\|_K^2 = \min \left\{ \sum_{j \in \mathbf{N}_n} \frac{\|f_j\|^2}{\lambda_j} : f = \sum_{j \in \mathbf{N}_n} f_j, f_\ell \in \mathcal{H}_{K_\ell}, \ell \in \mathbf{N}_n \right\}.$$

Thus, the lemma follows from the following fact.

Lemma 26 *If $r > 0$, $p := 1 + \frac{1}{r}$, and $\{a_j : j \in \mathbf{N}_n\} \subset \mathbb{R}$ then*

$$\min \left\{ \left(\sum_{j \in \mathbf{N}_n} \frac{a_j^2}{\lambda_j} \right)^{\frac{1}{2}} : \lambda_\ell \geq 0, \ell \in \mathbf{N}_n, \sum_{j \in \mathbf{N}_n} \lambda_j^r \leq 1 \right\} = \left(\sum_{j \in \mathbf{N}_n} |a_j|^{\frac{2}{p}} \right)^{\frac{p}{2}}$$

and the equality occurs for $\sum_{j \in \mathbf{N}_n} |a_j| > 0$ at

$$\tilde{\lambda}_j := \frac{|a_j|^{\frac{2}{r+1}}}{\left(\sum_{j \in \mathbf{N}_n} |a_j|^{\frac{2r}{r+1}} \right)^{\frac{1}{r}}}. \quad (36)$$

PROOF. This fact follows from Hölder inequality. To this end, we let $q = r + 1$ so that $\frac{1}{p} + \frac{1}{q} = 1$ and, so, we have that

$$\begin{aligned} \sum_{j \in \mathbf{N}_n} |a_j|^{\frac{2r}{q}} &= \sum_{j \in \mathbf{N}_n} \frac{|a_j|^{\frac{2r}{q}}}{\lambda_j^{\frac{r}{q}}} \lambda_j^{\frac{r}{q}} \\ &\leq \left(\sum_{j \in \mathbf{N}_n} \frac{|a_j|^{\frac{2rp}{q}}}{\lambda_j^r} \right)^{\frac{1}{p}} \left(\sum_{j \in \mathbf{N}_n} \lambda_j^r \right)^{\frac{1}{q}} \\ &= \left(\sum_{j \in \mathbf{N}_n} \frac{a_j^2}{\lambda_j} \right)^{\frac{1}{p}} \left(\sum_{j \in \mathbf{N}_n} \lambda_j^r \right)^{\frac{1}{q}} \leq \left(\sum_{j \in \mathbf{N}_n} \frac{a_j^2}{\lambda_j} \right)^{\frac{1}{p}}. \end{aligned}$$

For the choice (36) equality holds above, thereby completing the proof. □

The proof of Lemma 25 is completed. □

References

- A. Argyriou, C. A. Micchelli and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. *Proc. 18-th Annual Conference on Learning Theory (COLT'05)*, Bertinoro, Italy, June, 2005.
- N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 686: 337–404, 1950.
- J. P. Aubin. *Mathematical methods of game and economic theory*. Studies in Mathematics and its applications, Vol. 7, North-Holland, 1982.
- F. R. Bach, G. R. G. Lanckriet and M. I. Jordan. Multiple kernels learning, conic duality, and the SMO algorithm. *Proc. of the Int. Conf. on Machine Learning (ICML'04)* 2004.
- F. R. Bach, R. Thibaux and M. I. Jordan. Computing regularization paths for learning multiple kernels. *Advances in Neural Information Processing Systems*, 17, 2004.
- C. Bennett and R. Sharpley. *Interpolation of Operators*. Vol. 129, Pure and Appl. Math, Academic Press, Boston, 1988.
- J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization. Theory and Examples* CMS (Canadian Mathematical Society) Springer-Verlag, New York, 2000.
- O. Bousquet and A. Elisseeff. Stability and generalization. *J. of Machine Learning Research*, 2: 499–526, 2002.
- O. Bousquet and D. J. L. Herrmann. On the complexity of learning the kernel matrix. *Advances in Neural Information Processing Systems*, 15, 2003.
- O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1): 131–159, 2002.
- F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, 39 (1): 1–49, 2002.
- N. Cristianini, J. Shawe-Taylor, A. Elisseeff, J. Kandola. On kernel-target alignment *Advances in Neural Information Processing Systems*, 14, T. G. Dietterich, S. Becker, Z. Ghahramani (eds.), 2002.
- E. De Vito, L. Rosasco, A. Caponnetto, M. Piana, A. Verri. Some properties of regularized kernel methods. *J. of Machine Learning Research*, 5(Oct):1363–1390, 2004.
- T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13: 1–50, 2000.
- F. Girosi. An Equivalence Between Sparse Approximation and Support Vector Machines. *Neural Computation*, 10 (6): 1455–1480, 1998.
- T. Graepel. Kernel matrix completion by semi-definite programming. *Proc. of ICANN*, pages 694–699, 2002.

- T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics, 2002.
- T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for support vector machines, *J. of Machine Learning Research*, 5, 1391–1415, 2004.
- M. Herbster. Learning Additive Models Online with Fast Evaluating Kernels. *Proc. of the The 14-th Annual Conference on Computational Learning Theory (COLT)*, pages 444–460, 2001.
- M. Herbster. Relative Loss Bounds and Polynomial-time Predictions for the K-LMS-NET Algorithm. *Proc. of the 15-th Int. Conference on Algorithmic Learning Theory*, October 2004.
- T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. MIT AI-Lab Technical Report, 1999.
- G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33: 82–95, 1971.
- G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M. I. Jordan. Learning the kernel matrix with semi-definite programming. In C. Sammut and A. Hoffmann (Eds.), *Proc. of the 19-th Int. Conf. on Machine Learning*, Sydney, Australia, Morgan Kaufmann, 2002.
- G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M. I. Jordan. Learning the kernel matrix with semi-definite programming. *J. of Machine Learning Research*, 5: 27–72, 2004.
- Y. Lee, Y. Kim, S. Lee and J.-Y. Koo. Structured Multicategory Support Vector Machine with ANOVA decomposition. Technical Report No. 743, Department of Statistics, The Ohio State University, October 2004.
- Y. Lin and H. H. Zhang. Component Selection and Smoothing in Smoothing Spline Analysis of Variance Models – COSSO. Institute of Statistics Mimeo Series 2556, NCSU, January 2003.
- O. L. Mangasarian. *Nonlinear Programming*. Classics in Applied Mathematics, SIAM, 1994.
- A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and its Applications*. Academic Press, San Diego, 1979.
- C. A. Micchelli. *Saturation Classes and Iterates of Operators*. PhD Thesis, Stanford University, 1969.
- C. A. Micchelli and M. Pontil. A function representation for learning in Banach spaces. *Proc. of the 17-th Annual Conf. on Learning Theory (COLT'04)*, Banff, Alberta, June 2004.
- C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17: 177–204, 2005.
- C. A. Micchelli, M. Pontil, Q. Wu, and D. X. Zhou. Error bounds for learning the kernel. Research Note 05/09, Dept of Computer Science, University College London, June, 2005.
- S. Mukherjee, P. Niyogi, T. Poggio, R. Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for empirical risk minimization. *Advances in Computational Mathematics*, to appear, 2004.

- C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. *Advances in Neural Information Processing Systems*, 15, S. Becker, S. Thrun, K. Obermayer (Eds.), MIT Press, Cambridge, MA, 2003.
- M. Pontil and A. Verri. Properties of support vector machines. *Neural Computation*, 10: 955–974, 1998.
- R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.
- H. L. Royden. *Real Analysis*. Macmillan Publishing Company, New York, 3rd edition, 1988.
- I. J. Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, 39(4): 811–841, 1938.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, USA, 2002.
- C. Scovel and I. Steinwart. Fast rates for support vector machines. Preprint, 2004.
- J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- S. Smale, and D. X. Zhou. Estimating the approximation error in learning theory, *Anal. Appl.*, 1: 1–25, 2003.
- D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20: 1191–1199, 1999.
- V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
- G. Wahba. *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.
- C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. *Advances in Neural Processing Systems* 8: 598–604, D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (eds.), MIT Press, Cambridge, MA, 1996.
- Q. Wu, Y. Ying and D. X. Zhou. Multi-kernel regularization classifiers. *Preprint*, City University of Hong Kong, 2004.
- Y. M. Ying and D. X. Zhou. Learnability of Gaussians with flexible variances. Preprint, City University of Hong Kong, 2004.
- T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Statis.*, 32: 56–85, 2004.
- T. Zhang. On the dual formulation of regularized linear systems with convex risks. *Machine Learning*, 46: 91–129, 2002.
- Z. Zhang, D.-Y. Yeung and J. T. Kwok. Bayesian inference for transductive learning of kernel matrix using the Tanner-Wong data augmentation algorithm. *Proc. 21-st Int. Conf. Machine Learning (ICML-2004)*, pages 935-942, Banff, Alberta, Canada, July 2004.
- D. X. Zhou. The covering number in learning theory. *J. of Complexity*, 18: 739–767, 2002.

Analysis of Variance of Cross-Validation Estimators of the Generalization Error

Marianthi Markatou

Hong Tian

Department of Biostatistics

Columbia University

New York, NY 10032, USA

MM168@COLUMBIA.EDU

HT2031@COLUMBIA.EDU

Shameek Biswas

George Hripcsak

Department of Biomedical Informatics

Columbia University

New York, NY 10032, USA

SPB2003@COLUMBIA.EDU

GH13@COLUMBIA.EDU

Editor: David Madigan

Abstract

This paper brings together methods from two different disciplines: statistics and machine learning. We address the problem of estimating the variance of cross-validation (CV) estimators of the generalization error. In particular, we approach the problem of variance estimation of the CV estimators of generalization error as a problem in approximating the moments of a statistic. The approximation illustrates the role of training and test sets in the performance of the algorithm. It provides a unifying approach to evaluation of various methods used in obtaining training and test sets and it takes into account the variability due to different training and test sets. For the simple problem of predicting the sample mean and in the case of smooth loss functions, we show that the variance of the CV estimator of the generalization error is a function of the moments of the random variables $Y = \text{Card}(S_j \cap S_{j'})$ and $Y^* = \text{Card}(S_j^c \cap S_{j'}^c)$, where $S_j, S_{j'}$ are two training sets, and $S_j^c, S_{j'}^c$ are the corresponding test sets. We prove that the distribution of Y and Y^* is hypergeometric and we compare our estimator with the one proposed by Nadeau and Bengio (2003). We extend these results in the regression case and the case of absolute error loss, and indicate how the methods can be extended to the classification case. We illustrate the results through simulation.

Keywords: cross-validation, generalization error, moment approximation, prediction, variance estimation

1. Introduction

Progress in digital data acquisition and storage technology has resulted in the growth of very large databases. At the same time, interest has grown in the possibility of tapping these data and of extracting information from the data that might be of value to the owner of the database. A variety of algorithms have been developed to mine through these databases with the purpose of uncovering interesting characteristics of the data and generalizing the findings to other data sets.

One important aspect of algorithmic performance is the generalization error. Informally, the generalization error is the error an algorithm makes on cases that has never seen before. Thus, the generalization performance of a learning method relates to its prediction capability on the indepen-

dent test data. The assessment of the performance of learning algorithms is extremely important in practice because it guides the choice of learning methods.

The generalization error of a learning method can be easily estimated via either cross-validation or bootstrap. However, providing a variance estimate of the estimator of this generalization error is a more difficult problem. This is because the generalization error depends on the loss function involved, and the mathematics needed to analyze the variance of the estimator are complicated. An estimator of variance of the cross-validation estimator of the generalization error is proposed by Nadeau and Bengio (2003). In a later section of this paper we will discuss this estimator and compare it with the newly proposed estimator.

In this paper we address estimation of the variance of the cross validation estimator of the generalization error, using the method of moment approximation. The idea is simple. The cross validation estimator of the generalization error is viewed as a statistic. As such, it has a distribution. We then approximate the needed moments of this distribution in order to obtain an estimate of the variance. We present a framework that allows computation of the variance estimator of the generalization error for k fold cross validation, as well as the usual random set selection in cross validation. We address the problem of loss function selection and we show that for a general class of loss functions, the class of differentiable loss functions with certain tail behavior, and for the simple problem of prediction of the sample mean, the variance of the cross validation estimator of the generalization error depends on the expectation of the random variables $Y = \text{Card}(S_j \cap S_{j'})$ and $Y^* = \text{Card}(S_j^c \cap S_{j'}^c)$. Here $S_j, S_{j'}$ are two different training sets drawn randomly from the data universe and $S_j^c, S_{j'}^c$ are their corresponding test sets taken to be the complement of S_j and $S_{j'}$ with respect to the data universe. We then obtain variance estimators of the generalization error for the k -fold cross validation estimator, and extend the results to the regression case. We also indicate how the results can be extended to the classification case.

The paper is organized as follows. Section 2 introduces the framework and discusses existing literature on the problem of variance estimation of the cross validation estimators of the generalization error. Section 3 presents the moment approximation method for developing the new estimator. Section 4 presents computer experiments and compares our estimator with the estimator proposed by Nadeau and Bengio (2003). Section 5 presents discussion and conclusions.

2. Framework and Related Work

In what follows we describe the framework within which we will work.

2.1 The Framework and the Cross Validation Estimator of the Generalization Error

Let data X_1, X_2, \dots, X_n be collected such that the data universe, $Z_1^n = \{X_1, X_2, \dots, X_n\}$, is a set of independent, identically distributed observations which follow an unknown probability distribution, denoted by F . Let S represent a subset of size n_1 , $n_1 < n$, taken from Z_1^n . This subset of observations is called a training set; on the basis of a training set a rule is constructed. The test set contains all data that do not belong in S , that is the test set is the set $S^c = Z_1^n \setminus S$, the complement of S with respect to the data universe Z_1^n . Denote by n_2 the number of elements in a test set, $n_2 = n - n_1$, $n_2 < n$.

Let $L : \mathbb{R}^p \times \mathbb{R} \rightarrow \mathbb{R}$ be a function, and assume that Y is a target variable and $\hat{f}(x)$ is a decision rule. The function $L(Y, \hat{f}(X))$ that measures the error between the target variable and the prediction rule is called a loss function.

As an example, consider the estimation of the sample mean. In this problem the learning algorithm uses $\hat{f}(x) = \frac{1}{n_1} \sum_{i=1}^{n_1} X_i = \bar{X}_{S_j}$ as a decision rule and $L(\bar{X}_{S_j}, X_i) = (\bar{X}_{S_j} - X_i)^2$, $X_i \in S_j^c$, the square error loss, as a loss function. Other typical choices of the loss function include the absolute error loss, $|\bar{X}_{S_j} - X_i|$ and the 0 – 1 loss function mainly used in classification.

Our results take into account the variability in both training and test sets. The variance estimate of the cross validation estimator of the generalization error can be computed under the following cross validation schemes. The first is what we term as *complete random selection*. When this form of cross validation is used to compute the estimate of the generalization error of a learning method, the training sets, and hence the test sets, are randomly selected from the available data universe. In the *nonoverlapping test set selection* case, the data universe is divided into k nonoverlapping data subsets. Each data subset is then used as a test set, with the remaining data acting as a training set. This is the case of k-fold cross validation.

We now describe in detail the cross validation estimator of the generalization error whose variance we will study. This estimator is constructed under the complete random selection case.

Let A_j be a random set of n_1 distinct integers from $\{1, 2, \dots, n\}$, $n_1 < n$. Let $n_2 = n - n_1$ be the size of the corresponding complement set. Note here that n_2 is a fixed number and that $Card(A_j) = n_1$ is fixed. Let A_1, A_2, \dots, A_J be random index sets sampled independently of each other and denote by A_j^c , the complement of A_j , $j = 1, 2, \dots, J$. Denote also by $S_j = \{X_l : l \in A_j\}$, $j = 1, 2, \dots, J$. This is the training set obtained by subsampling Z_1^n according to the random index set A_j . Then the corresponding test set is $S_j^c = \{X_l : l \in A_j^c\}$. Now define $L(j, i) = L(S_j, X_i)$, where L is a loss function. Notice that L is defined by its dependence on the training set S_j and the test set S_j^c . This dependence on the training and test sets is through the statistics that are computed using the elements of these sets. The usual average test set error is then

$$\hat{\mu}_j = \frac{1}{n_2} \sum_{i \in S_j^c} L(j, i), \tag{2.1}$$

The cross validation estimator we will study is defined as

$${}_{n_1}^{n_2} \hat{\mu}_J = \frac{1}{J} \sum_{j=1}^J \hat{\mu}_j. \tag{2.2}$$

This version of the cross validation estimator of the generalization error depends on the value of J , the size of the training and test sets and the size of the data universe. The estimator has been studied by Nadeau and Bengio (2003). These authors provided two estimators of the variance of ${}_{n_1}^{n_2} \hat{\mu}_J$. In the next section we review briefly the estimators presented by Nadeau and Bengio (2003) as well as other work on this subject. In a later section we will see that, when J is chosen appropriately, then the Nadeau and Bengio (2003) estimator is close to and performs similarly with the moment approximation estimator in some of the cases we study.

2.2 Related Work

Related literature for the problem of estimating the variance of the generalization error includes work by McLachlan (1972, 1973, 1974, 1976) and work by Nadeau and Bengio (2003) and Bengio and Grandvalet (2004). Here, we briefly review this work.

Let $S_{\hat{\mu}_j}^2 = \frac{1}{J-1} \sum_{j=1}^J (\hat{\mu}_j - {}_{n_1}^{n_2} \hat{\mu}_J)^2$ be the sample variance of $\hat{\mu}_j$, $j = 1, 2, \dots, J$. Then Nadeau and Bengio (2003) show that

$$E(S_{\hat{\mu}_j}^2) = \frac{\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J)}{\left(\frac{1}{J} + \frac{\rho}{1-\rho}\right)}, \quad (2.3)$$

where ρ is the correlation between $\hat{\mu}_j$ and $\hat{\mu}_j$. Therefore, if ρ is known,

$$\left(\frac{1}{J} + \frac{\rho}{1-\rho}\right)S_{\hat{\mu}_j}^2, \quad (2.4)$$

is an unbiased estimator of the $\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J)$. Nadeau and Bengio (2003) observe that this estimator depends on the correlation ρ between the different $\hat{\mu}_j$ s which is difficult to estimate. Thus, they propose an approximation to the correlation, $\hat{\rho} = \frac{n_2}{n}$, where n_2 is the cardinality of the test set. The final estimator of the variance of ${}_{n_1}^{n_2}\hat{\mu}_J$ is given as

$$\left(\frac{1}{J} + \frac{n_2}{n_1}\right)S_{\hat{\mu}_j}^2. \quad (2.5)$$

Nadeau and Bengio (2003) note that the above suggested estimator is simple but it may have a positive or negative bias with respect to the actual $\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J)$. That is, it will tend to overestimate or underestimate $\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J)$ according to whether $\hat{\rho} = \frac{n_2}{n} > \rho$ or $\hat{\rho} < \rho$. Therefore, this estimator is not exactly unbiased.

Nadeau and Bengio (2003) also suggested another estimator of the variance of the cross-validation estimator of the generalization error. This estimator is unbiased but overestimates the $\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J)$. It is computed as follows. Let n be the size of the data universe and assume, without loss of generality, that n is even. Randomly split the data set into two, equal size, data subsets. Then compute the cross-validation estimator of the generalization error on these two data subsets. Notice that, the size of the training set is now $n'_1 = \lfloor \frac{n}{2} \rfloor - n_2 < n_1$, smaller than the original size of the training set, but the test set size remains the same. Denote by $\hat{\mu}_1$ the estimator ${}_{n'_1}^{n_2}\hat{\mu}_J$ computed on the first data subset and $\hat{\mu}_2$ the estimator ${}_{n'_1}^{n_2}\hat{\mu}_J$ computed on the second data subset. To obtain an estimator of the variance of the cross validation estimator of the generalization error compute the sample variance of $\hat{\mu}_1$ and $\hat{\mu}_2$. The splitting process can be repeated M times and Nadeau and Bengio(2003) recommend $M = 10$. The proposed unbiased estimator is then given as

$$\frac{1}{2M} \sum_{m=1}^M (\hat{\mu}_{1,m} - \hat{\mu}_{2,m})^2. \quad (2.6)$$

This is an unbiased estimator of the $\text{Var}({}_{n'_1}^{n_2}\hat{\mu}_J)$.

Bengio and Grandvalet (2004) showed that there does not exist any unbiased and universal estimator of the variance of k-fold cross-validation that is valid under all distributions. Here, we derive estimators of the variance of the k-fold cross validation estimator of the generalization error that are almost unbiased. However, we also notice that our estimators do depend on the distribution of the errors and on the knowledge of the learning algorithm.

In a series of impressive papers McLachlan addressed the problem of estimation of the variance of the errors of misclassification of the linear discriminant function by developing a technique for deriving asymptotic expansions of the variances of the errors of misclassification of Anderson's classification statistic. McLachlan also established an asymptotic expansion of the expectation of the estimated error rate in discriminant analysis and obtained the distributions of the conditional error

rate and risk associated with Anderson’s classification statistic in the context of the two-population discrimination problem. These derivations were carried out under the assumption of normality for the population distribution.

Our work has similarities with the work by McLachlan in the sense that we derive approximations to the moments of the distribution of the cross validation estimator of the generalization error and use these to obtain a variance estimator. However, we do not assume normality of the underlying mechanism that generated the data.

In what follows, we first present the method of moment approximation for obtaining an estimator of $Var(\hat{\mu}_J)$. We then study the performance of this estimator and compare it with the Nadeau and Bengio (2003) estimator.

3. Moment Approximation Estimator for $Var(\hat{\mu}_J)$

Recall that $\hat{\mu}_J = \frac{1}{J} \sum_{j=1}^J \hat{\mu}_j = \frac{1}{J} \sum_{j=1}^J (\frac{1}{n_2} \sum_{i \in S_j^c} L(j, i))$. Therefore $\hat{\mu}_J$ is a statistic. An estimator of $Var(\hat{\mu}_J)$ can thus be obtained by approximating the moments of the statistic $\hat{\mu}_J$. A simple calculation shows that

$$Var(\hat{\mu}_J) = \frac{1}{J^2} \sum_{j=1}^J Var(\hat{\mu}_j) + \frac{1}{J^2} \sum_{j \neq j'} Cov(\hat{\mu}_j, \hat{\mu}_{j'}). \tag{3.1}$$

From the formula we see that if we can approximate the two terms of (3.1) then we can obtain an estimator for the variance of $\hat{\mu}_J$. To achieve this goal, we need to estimate $E(\hat{\mu}_j)$, $E(\hat{\mu}_j^2)$ and $E(\hat{\mu}_j \hat{\mu}_{j'})$. In the following sections we will develop the theory that allows us to obtain the needed moment approximations. To illustrate the methodology clearly we treat separately the case of simple mean estimation and the regression case. We further treat separately the case where the loss function is differentiable from the case of non-differentiable loss functions.

3.1 The Sample Mean Case

We start by analyzing the case of the sample mean. Here, the loss function L depends on S_j through the statistics \bar{X}_{S_j} , the sample mean computed using the elements of S_j , and on S_j^c by elements $X_i \in S_j^c$. One of the reasons for presenting the sample mean case separately is because it illustrates clearly the contribution towards the estimator of $Var(\hat{\mu}_J)$ that is due to the variability among the different training and test sets. A second reason in favor of this case is because, under square error loss, we obtain a “golden standard” against which we can compare the new empirically computed variance estimator and the Nadeau and Bengio (2003) estimator. This “golden standard” is the exact theoretical value of the $Var(\hat{\mu}_J)$. The obtained results show that the estimator of the variance of the cross validation estimator of the generalization error of the algorithms that use differentiable functions of the mean as loss functions, depends on the expectation of the random variables $Y = Card(S_j \cap S_{j'})$ and $Y^* = Card(S_j^c \cap S_{j'}^c)$.

Let the loss function $L(j, i) = L(\bar{X}_{S_j}; X_i)$ be differentiable. Below we list the conditions under which our theory holds.

Assumption 1. The distribution of $L(\bar{X}_{S_j}, X_i)$ does not depend on the particular realization of S_j and i .

Assumption 2. The loss function L as a function of \bar{X}_{S_j} is such that its first four derivatives with respect to the first argument exist for all values of the variable that belongs in I , where I is an interval such that $P(v \in I) = 1$, and v indicates the first argument of the loss function.

Assumption 3. The fourth derivative of L is such that $|L^{(iv)}(\bar{X}_{S_j}; X_i)| \leq M(X_i)$, $E[M(X_i)] < \infty$.

Assumption 1 is also used by Nadeau and Bengio (2003, p. 244). Assumptions 2 and 3 are standard in the literature where approximations to the moments of a continuous, real function of the mean are discussed. See, for example Cramer (1946), Lehman (1991) and Bickel and Doksum (2001). The boundedness of the fourth or some higher derivative is necessary for proposition 3.1 to hold.

Alternative conditions where stronger assumptions on the distributions of the data X_i and weaker conditions on the function L are imposed exist in the literature (Khan (2004)). Here L is a loss function and it seems reasonable to assume boundedness on some of its higher derivatives.

Proposition 3.1 offers an approximation of the expectation of $L(\bar{X}_{S_j}, X_i)$.

Proposition 3.1 Let X_1, X_2, \dots, X_n be independent, identically distributed random variables such that $E(X_i) = \mu$, $Var(X_i) = \sigma^2$ and finite fourth moment. Suppose that L satisfies assumptions 1, 2 and 3. Then

$$E[L(\bar{X}_{S_j}; X_i)] = E[L(\mu, X_i)] + \frac{\sigma^2}{2n_1} E[(L''(\mu, X_i))] + O\left(\frac{1}{n_1^2}\right),$$

where the remainder R_n is such that $E(R_n)$ is $O\left(\frac{1}{n_1^2}\right)$, that is, there exists n_0 and $A < \infty$ such that $E(R_n) < \frac{A}{n_1^2}$, $\forall n > n_0$ and all μ . The prime indicates derivative with respect to the first argument of L .

Proof: We will use a conditional expectation argument. Write

$$E[L(\bar{X}_{S_j}; X_i)] = E_{S_j, i} \{ E_{Z_1^n} [L(\bar{X}_{S_j}; X_i) | S_j, i] \}, \tag{3.2}$$

$j = 1, 2, \dots, J$ and i indicates X_i and is such that $i \in S_j^c$.

Now expand $L(\bar{X}_{S_j}; X_i)$ with respect to \bar{X}_{S_j} around the mean μ to obtain:

$$\begin{aligned} L(\bar{X}_{S_j}; X_i) &= L(\mu, X_i) + L'(\mu, X_i)(\bar{X}_{S_j} - \mu) + \frac{1}{2} L''(\mu, X_i)(\bar{X}_{S_j} - \mu)^2 \\ &+ \frac{1}{6} L'''(\mu, X_i)(\bar{X}_{S_j} - \mu)^3 + \frac{1}{24} L^{(iv)}(\mu^*, X_i)(\bar{X}_{S_j} - \mu)^4. \end{aligned} \tag{3.3}$$

Denote by

$$R_n = L^{(iv)}(\mu^*, X_i)(\bar{X}_{S_j} - \mu)^4$$

and

$$E_{Z_1^n} \{ R_n | S_j, i \} = E_{Z_1^n} \{ L^{(iv)}(\mu^*, X_i)(\bar{X}_{S_j} - \mu)^4 | S_j, i \}, \tag{3.4}$$

and since by assumption 1 the distribution of $L^{(iv)}(\mu^*, X_i)(\bar{X}_{S_j} - \mu)^4$ does not depend on the particular realization of S_j and i , we obtain

$$E_{S_j, i} \{ E_{Z_1^n} [L^{(iv)}(\mu^*, X_i)(\bar{X}_{S_j} - \mu)^4 | S_j, i] \} = E[L^{(iv)}(\mu^*, X_i)] E(\bar{X}_{S_j} - \mu)^4 \leq M \cdot E(\bar{X}_{S_j} - \mu)^4.$$

This is because by assumption 3 we have $E[L^{(iv)}(\mu^*, X_i)] \leq E[M(X_i)] < \infty$. Now Lemma A.5 of the appendix guarantees that $E(\bar{X}_{S_j} - \mu)^4$ is of order $1/n_1^2$. Thus, taking expectations in (3.3) and using (3.4) we obtain:

$$\begin{aligned} E[L(\bar{X}_{S_j}; X_i)] &= E_{S_j, i} \{ E_{Z_1^n} [L(\mu, X_i) | S_j, i] \} + E_{S_j, i} \{ E_{Z_1^n} [L'(\mu, X_i)(\bar{X}_{S_j} - \mu) | S_j, i] \} \\ &+ E_{S_j, i} \{ E_{Z_1^n} [\frac{1}{2} L''(\mu, X_i)(\bar{X}_{S_j} - \mu)^2 | S_j, i] \} \\ &+ E_{S_j, i} \{ E_{Z_1^n} [\frac{1}{6} L'''(\mu, X_i)(\bar{X}_{S_j} - \mu)^3 | S_j, i] \} + O(\frac{1}{n_1^2}). \end{aligned}$$

By assumption 1 the distribution of $L(\mu, X_i)$ does not depend on the particular realization of S_j and X_i . Thus

$$E_{S_j, i} \{ E_{Z_1^n} [L(\mu, X_i) | S_j, i] \} = E_{Z_1^n} [L(\mu, X_i)].$$

Similar to the above arguments produce the approximation to the first moment given by

$$E[L(\bar{X}_{S_j}; X_i)] = E[L(\mu, X_i)] + \frac{\sigma^2}{2n_1} E[(L''(\mu, X_i))] + O(\frac{1}{n_1^2}).$$

Remark 1: Note that we do not impose distributional assumptions on the data. The only condition imposed is that samples come from distributions for which the fourth moment is finite. Many of the standard families of distributions satisfy this condition.

Remark 2: The requirement of the finiteness of the fourth moment for proposition 3.1 to hold implies limitations on the data sets on which this estimator can be computed. For example, it may be inappropriate to apply these methods to data sets which involve large variations, such as those from insurance and finance. On the other hand, the results apply to some thick tail distributions, such as the t -distribution with 5 or more degrees of freedom. The t_5 -distribution, for example, is a thick tail distribution, for which the fourth moment exists.

The following proposition approximates the variance of the loss $L(\bar{X}_{S_j}, X_i)$.

Proposition 3.2 Let assumptions 1, 2 and 3 hold. If in addition the fourth derivative of $L^2(\bar{X}_{S_j}, X_i)$ is bounded, then

$$Var[L(\bar{X}_{S_j}; X_i)] = Var[L(\mu, X_i)] + \frac{\sigma^2}{n_1} \{ E[(L'(\mu, X_i))^2] + Cov(L(\mu, X_i), L''(\mu, X_i)) \} + O(1/n_1^2),$$

where the remainder term is $O(\frac{1}{n_1^2})$.

Proof: To obtain an expansion of the variance of $L(\bar{X}_{S_j}; X_i)$ apply proposition 1 to the function $L^2(\bar{X}_{S_j}; X_i)$ using the fact that

$$\begin{aligned} [L^2(\mu, X_i)]'' &= \frac{\partial^2}{\partial \mu^2} [L^2(\mu, X_i)] \\ &= 2(L'(\mu, X_i))^2 + 2L(\mu, X_i)L''(\mu, X_i). \end{aligned} \tag{3.5}$$

Then substituting the expansion for $L(\bar{X}_{S_j}, X_i)$ and using formula (3.5), proposition 1 and the formula of conditional variance we obtain:

$$\text{Var}[L(\bar{X}_{S_j}, X_i)] = \text{Var}[L(\mu, X_i)] + \frac{\sigma^2}{n_1} \{E[(L'(\mu, X_i))^2] + \text{Cov}(L(\mu, X_i), L''(\mu, X_i))\} + O(1/n_1^2).$$

To prove the above two propositions we use a series of lemmas that guarantee the rate of the remainder term. These lemmas are presented in the appendix.

We now present a theoretical example that verifies the approximations presented in propositions 1 and 2.

Example. Assume that $L(\bar{X}_{S_j}, X_i) = (\bar{X}_{S_j} - X_i)^2$, the square error loss that is widely used. An exact calculation of the expectation of $(\bar{X}_{S_j} - X_i)^2$ produces

$$E\{L(\bar{X}_{S_j}, X_i)\} = \text{Var}(\bar{X}_{S_j}) + \text{Var}(X_i) = \sigma^2 + \frac{\sigma^2}{n_1}.$$

On the other hand, if proposition 3.1 is used, we obtain:

$$E[L(\bar{X}_{S_j}, X_i)] = E(X_i - \mu)^2 + \frac{\sigma^2}{n_1} = \sigma^2 + \frac{\sigma^2}{n_1},$$

and the two formulas coincide. Notice that in the case of square error loss, the second derivative of the loss, with respect to μ , is bounded. The terms of order $1/n_1^2$ do not enter the formula as all higher order than two derivatives of the quadratic loss are 0. Thus, the approximation formula agrees with the exact computation.

We next turn to the variance formula. The exact computation is based on the formula

$$\text{Var}[L(\bar{X}_{S_j}, X_i)] = E_{S_j, i} \{ \text{Var}_{Z_1^n} [(\bar{X}_{S_j} - X_i)^2 | S_j, i] \} + \text{Var}_{S_j, i} \{ E_{Z_1^n} [(\bar{X}_{S_j} - X_i)^2 | S_j, i] \}. \quad (3.6)$$

Using this formula we obtain the exact variance as

$$\text{Var}[L(\bar{X}_{S_j}, X_i)] = 2\sigma^4 + \frac{4\sigma^4}{n_1} + \frac{2\sigma^4}{n_1^2}. \quad (3.7)$$

Using the formula given in proposition 3.2 we obtain that the approximate variance is

$$\text{Var}[L(j, i)] = 2\sigma^4 + \frac{4\sigma^4}{n_1} + O\left(\frac{1}{n_1^2}\right). \quad (3.8)$$

Comparing these two formulas we see that the variance approximation formula identifies all first order terms.

The following proposition establishes the approximation formula for the covariance terms that enter the computation of the variance of the cross validation estimators of the generalization error.

Proposition 3.3 Let $S_j, S_{j'}$ be two training sets drawn independently and at random from the data universe Z_1^n , and $S_j^c, S_{j'}^c$ the corresponding test sets. Let $X_i \in S_j^c, X_{i'} \in S_{j'}^c, D = S_j \cap S_{j'}$ and $Y = \text{Card}(D)$. Then, if $i \neq i'$

$$\text{Cov}[L(\bar{X}_{S_j}, X_i), L(\bar{X}_{S_{j'}}, X_{i'})] = \frac{\sigma^2}{n_1^2} E(Y) (E[L'(\mu, X_i)])^2 - \frac{\sigma^4}{4n_1^2} (E[L''(\mu, X_i)])^2 + O\left(\frac{1}{n_1^2}\right).$$

If $i = i'$,

$$\begin{aligned} Cov[L(\bar{X}_{S_j}, X_i), L(\bar{X}_{S_{j'}}, X_{i'})] &= Var(L(\mu, X_i)) + \frac{\sigma^2}{n_1} \{E[L(\mu, X_i)L''(\xi, i)] \\ &- E[L(\mu, X_i)]E[L''(\mu, X_i)]\} + \frac{\sigma^2}{n_1^2} E(Y)E[L'(\mu, X_i)]^2 \\ &- \frac{\sigma^4}{4n_1^2} \{E[L''(\mu, X_i)]\}^2 + O\left(\frac{1}{n_1^2}\right), \end{aligned}$$

where $E(Y)$ is the expectation of the random variable Y with respect to its distribution.

This proposition indicates that the variability due to random sampling of the training sets S_j is quantified by the expectation of the random variable $Y = Card(S_j \cap S_{j'})$, $j \neq j'$, $j, j' \in 1, 2, \dots, J$. Since $S_j, S_{j'}$ are random sets of n_1 elements, Y is such that $\max(0, 2n_1 - n) \leq Y \leq n_1$.

An additional random variable that enters the variance estimator of the cross validation estimator of the generalization error is $Y^* = Card(S_j^c \cap S_{j'}^c)$, the cardinality of the intersection of two different test sets. The following two lemmas derive the distribution of these two random variables.

Lemma 3.1 Let S_j and $S_{j'}$ be random sets of n_1 distinct elements from Z_1^n and let $Y = Card(S_j \cap S_{j'})$, $\max(0, 2n_1 - n) \leq Y \leq n_1$. Then, the distribution of Y is

$$P(Y = y) = \frac{\binom{n_1}{y} \binom{n-n_1}{n_1-y}}{\binom{n}{n_1}},$$

a hypergeometric distribution.

Proof. We model the problem as the following $2 \times n$ table.

k	1	2	3	...	n	Total
S_j	0	1	1	...	0	n_1
$S_{j'}$	1	0	1	...	0	n_1
	a_1	a_2	a_3	...	a_n	$2n_1$

In the table we indicate whether the k th component of Z_1^n is sampled into the training set S_j or $S_{j'}$ by 1, otherwise we indicate it by 0. Denote by a_k the sum of the indicators for the k th component in the population Z_1^n over S_j and $S_{j'}$. Then

$$\begin{cases} a_1 + a_2 + \dots + a_n = 2n_1 \\ 0 \leq a_i \leq 2 \end{cases}, i = 1, \dots, n.$$

Now, $P(Y = y)$ is equivalent to $P(\#\{a_i = 2\})$, $i = 1, \dots, n$. Given $Y = y$, the number of $\{a_i = 1\}$ is $2n_1 - 2y$ and the number of $\{a_i = 0\}$ is $n - 2n_1 + y$. Since none of these three numbers could be negative, we obtain the domain of Y as $\max(0, 2n_1 - n) \leq Y \leq n_1$. Recall also that $S_j, S_{j'}$ are sampled independently and each contains n_1 elements. Given $Y = y$, the distribution of the column totals is fixed; that is a_i can only take the values 0, 1 or 2. The number of different tables with the same column totals is then $\binom{n}{y} \binom{n-y}{n_1-y} \binom{n-n_1}{n_1-y}$. and hence

$$P(Y = y) = \frac{\binom{n}{y} \binom{n-y}{n_1-y} \binom{n-n_1}{n_1-y}}{\binom{n}{n_1} \binom{n}{n_1}} = \frac{\binom{n_1}{y} \binom{n-n_1}{n_1-y}}{\binom{n}{n_1}},$$

the hypergeometric distribution.

Lemma 3.2 Let S_j and $S_{j'}$ be two training sets and S_j^c and $S_{j'}^c$ are their corresponding test sets. Let $Y^* = \text{Card}(S_j^c \cap S_{j'}^c)$, $0 \leq Y^* \leq n - n_1$. Then

$$P(Y^* = y) = \frac{\binom{n_1}{y-n+2n_1} \binom{n-n_1}{n-n_1-y}}{\binom{n}{n_1}} = \frac{\binom{n_2}{n_2-y} \binom{n-n_2}{n-n_2-(n_2-y)}}{\binom{n}{n-n_2}}.$$

Proof. From the proof of lemma 3.1 $P(Y^* = y) = P(\#\{a_i = 0\})$, $\{i = 1, \dots, n\}$. Moreover, $Y^* = n - 2n_1 + Y$. Then, the result follows.

Theorem 3.1 provides the estimator of the variance of $\frac{n_2}{n_1} \hat{\mu}_J$. We first state the theorem.

Theorem 3.1. The variance of the estimator of the generalization error $\frac{n_2}{n_1} \hat{\mu}_J$ is given as

$$\text{Var}(\frac{n_2}{n_1} \hat{\mu}_J) = \frac{1}{J} \text{Var}(\hat{\mu}_j) + \frac{J-1}{J} \text{Cov}(\hat{\mu}_j, \hat{\mu}_{j'}),$$

where

$$\begin{aligned} \text{Var}(\hat{\mu}_j) &= \frac{1}{n_2} [\text{Var}[L(\mu, X_i)] + \frac{\sigma^2}{n_1} \{E[(L'(\mu, X_i))^2] + \text{Cov}(L(\mu, X_i), L''(\mu, X_i))\}] \\ &\quad + \frac{n_2 - 1}{n_2} \frac{\sigma^2}{n_1} \{E[L'(\mu, X_i)]\}^2 + O(1/n_1^2), \\ \text{Cov}(\hat{\mu}_j, \hat{\mu}_{j'}) &= (1 - \frac{E(Y^*)}{n_2^2}) \left[\frac{\sigma^2}{n_1^2} E(Y) (E[L'(\mu, X_i)])^2 - \frac{\sigma^4}{4n_1^2} (E[L''(\mu, X_i)])^2 + O(\frac{1}{n_1^2}) \right] \\ &\quad + \frac{E(Y^*)}{n_2^2} \left[\text{Var}(L(\mu, X_i)) + \frac{\sigma^2}{n_1} \{E[L(\mu, X_i)L''(\mu, X_i)] - E[L(\mu, X_i)]E[L''(\mu, X_i)]\} \right] \\ &\quad + \frac{\sigma^2}{n_1^2} E(Y) E[L'(\mu, X_i)]^2 - \frac{\sigma^4}{4n_1^2} \{E[L''(\mu, X_i)]\}^2 + O(\frac{1}{n_1^2}), \end{aligned}$$

where $\mu = E_{Z_1^n} X_i$, $\sigma^2 = \text{Var}_{Z_1^n}(X_i)$.

The above formulas indicate clearly the dependence of $\text{Var}(\frac{n_2}{n_1} \hat{\mu}_J)$ on the first moment of the random variables Y, Y^* . Since the distribution of Y and Y^* is known, we can substitute $E(Y), E(Y^*)$ by their corresponding values and simplify the above expressions. Because the distribution of Y, Y^* is hypergeometric $E(Y) = \frac{n_2^2}{n}$ and $E(Y^*) = \frac{n_2^2}{n}$. Then

$$\begin{aligned} \text{Cov}(\hat{\mu}_j, \hat{\mu}_{j'}) &= (1 - \frac{1}{n}) \left[\frac{\sigma^2}{n} (E[L'(\mu, X_i)])^2 - \frac{\sigma^4}{4n_1^2} (E[L''(\mu, X_i)])^2 + O(\frac{1}{n_1^2}) \right] \\ &\quad + \frac{1}{n} \left[\text{Var}(L(\mu, X_i)) + \frac{\sigma^2}{n_1} \{ \text{Cov}(L(\mu, X_i), L''(\mu, X_i)) \} \right] \\ &\quad + \frac{\sigma^2}{n} E[L'(\mu, X_i)]^2 - \frac{\sigma^4}{4n_1^2} (E[L''(\mu, X_i)])^2 + O(\frac{1}{n_1^2}). \end{aligned}$$

The final estimator of the variance of ${}_{n_1}^{n_2}\hat{\mu}_J$ is a plug-in estimator and it can be computed using theorem (3.1). We need to replace the unknown population mean μ and population variance σ^2 by their estimators, the sample mean and sample variance respectively. If it is not convenient to compute the sample variance and mean based on the data universe we may compute \bar{X}_{S_j} and, if there are many different training sets, take as an estimator of the sample mean $\bar{X} = \frac{1}{J} \sum_{j=1}^J \bar{X}_{S_j}$. Moreover, $\hat{\sigma}_j^2 = \frac{1}{n_1-1} \sum_{l=1}^{n_1} (X_l - \bar{X}_{S_j})^2$, thus the variance estimate of the population variance will be $\hat{\sigma}^2 = \frac{1}{J} \sum_{j=1}^J \hat{\sigma}_j^2$.

Example. In the case of square error loss the approximations to the variance of $\hat{\mu}_j$ and the $Cov(\hat{\mu}_j, \hat{\mu}_{j'})$ are given as:

$$Var(\hat{\mu}_j) = \frac{1}{n_2} Var[(X_i - \mu)^2] + \frac{4\sigma^4}{n_1 n_2} = \frac{1}{n_2} E[(x_i - \mu)^4] - \frac{\sigma^4}{n_2} + \frac{4\sigma^4}{n_1 n_2}, \quad (3.9)$$

$$Cov(\hat{\mu}_j, \hat{\mu}_{j'}) = (1 - \frac{1}{n}) (-\frac{\sigma^4}{n_1^2}) + \frac{1}{n} (\frac{4\sigma^4}{n} - \frac{\sigma^4}{n_1^2} + Var[(X_i - \mu)^2]). \quad (3.10)$$

If the data are from a $N(0, \sigma^2)$ then the moment approximation estimator of the variance of ${}_{n_1}^{n_2}\hat{\mu}_J$ is given by

$$\hat{\sigma}^4 \left\{ \frac{2(n_1 + 2)}{n_1 n_2} \frac{1}{J} + \left(\frac{J-1}{J} \right) \left[\frac{2(n+2)}{n^2} - \frac{1}{n_1^2} \right] \right\},$$

where $\hat{\sigma}$ is the sample standard deviation. Thus the estimator of the variance ${}_{n_1}^{n_2}\hat{\mu}_J$ is a multiple of the sample variance and the multiplication factor indicates the dependence of the estimator on n_1, n_2 and n .

Variance estimator of the k-fold CV estimator of the generalization error.

Here we present a variance estimator of the k-fold cross validation estimator of the generalization error of a learning algorithm. Notice that this is a special case of theorem 3.1. In k-fold cross validation the data universe is divided into k different non-overlapping test sets, each of which contains $\frac{n}{k}$ elements. The number of elements n_1 , in any given training set, is then $n - \frac{n}{k} = \frac{(k-1)n}{k}$. Therefore, $Y = Card(S_j \cap S_{j'}) = \frac{(k-2)n}{k}$. Theorem 3.1 gives the approximations:

$$\begin{aligned} Var(\hat{\mu}_j) &= \frac{k}{n} [Var(L(\mu, X_i)) + \frac{\sigma^2}{n} \left(\frac{k}{k-1} \right) \{E[(L'(\mu, X_i))^2] + Cov(L(\mu, X_i), L''(\mu, X_i))\}] \\ &+ \frac{n-k}{n} \frac{\sigma^2}{n} \frac{k}{k-1} \{E[L'(\mu, X_i)]\}^2 + O(1/n_1^2), \end{aligned}$$

and

$$Cov(\hat{\mu}_j, \hat{\mu}_{j'}) = \frac{\sigma^2}{n} \frac{k(k-2)}{(k-1)^2} (E[L'(\mu, X_i)])^2 - \frac{\sigma^4}{4n^2} \left(\frac{k}{k-1} \right)^2 (E[L''(\mu, X_i)])^2 + O\left(\frac{1}{n_1^2}\right).$$

Therefore, the variance estimate can be computed using relation (3.1), where $Var(\hat{\mu}_j)$ and $Cov(\hat{\mu}_j, \hat{\mu}_{j'})$ are replaced by their estimates. These can be obtained by replacing μ, σ^2 by their sample estimates using data from the training sets.

Now assume that the loss function used is square error. In this case, $L'(\mu, x_i) = 2(\mu - x_i)$ and $L''(\mu, x_i) = 2$. The formulas then for the variance of $\hat{\mu}_j$ and the covariance between different $\hat{\mu}_j$'s simplify as follows:

$$\text{Var}(\hat{\mu}_j) = \frac{k}{n} \left\{ \text{Var}[(X_i - \mu)^2] + \frac{4\sigma^4}{n} \left(\frac{k}{k-1} \right) \right\}, \quad (3.11)$$

$$\text{Cov}(\hat{\mu}_j, \hat{\mu}_{j'}) = -\frac{\sigma^4}{n^2} \left(\frac{k}{k-1} \right)^2, \quad j \neq j'. \quad (3.12)$$

Then $\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J)$ can be estimated by using formula (3.1) and replacing σ^2 and $\text{Var}[(X_i - \mu)^2]$ by the sample variance and an appropriate sample estimate for $\text{Var}[(X_i - \mu)^2]$. The final approximation of the variance of ${}_{n_1}^{n_2}\hat{\mu}_J$ is then

$$\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J) = \frac{1}{n} \left\{ \text{Var}[(X_i - \mu)^2] \right\} + \frac{3k\sigma^4}{(k-1)n^2} = \frac{1}{n} E[(X_i - \mu)^4] - \frac{\sigma^4}{n} + \frac{3k\sigma^4}{(k-1)n^2}.$$

A simple estimator of $E[(X_i - \mu)^4]$ can be computed from the training sample by taking the sample version of the above expectation, $\frac{1}{n_1} \sum_{i \in S_j} (X_i - \bar{X}_{S_j})^4$. To illustrate, if we further assume a normal population then $\text{Var}[(X_i - \mu)^2] = 2\sigma^4$ and the variance estimator of ${}_{n_1}^{n_2}\hat{\mu}_J$ is given as

$$\frac{\hat{\sigma}^4}{n} \left(2 + \frac{3k}{n(k-1)} \right),$$

where $\hat{\sigma}$ is the sample standard deviation.

3.2 The Regression Case

The regression case is another case of fitting means. We consider here the problem of estimating the variance of the cross validation estimator of the generalization error ${}_{n_1}^{n_2}\hat{\mu}_J$ in the case of regression. Therefore the data are realizations of random variables (Y_i, X_i) , $i = 1, 2, \dots, n$ such that $E(Y_i|X_i) = x_i^T \beta$. Notice that the explanatory variables here are treated as fixed; this formulation is known as the fixed design case. The vector of unknown parameters β is usually estimated by least squares; denote by $\hat{\beta}$ the least square estimator of β . Then for a new observation $(y_i, x_i) \in S_j^c$ denote by $\hat{y}_{i, S_j} = x_i^T \hat{\beta}_{S_j}$, where $\hat{\beta}_{S_j}$ indicates the estimator of β computed by using the data in the training set S_j . The loss function L is then dependent on \hat{y}_{i, S_j} and y_i , that is $L(\hat{y}_{i, S_j}, y_i)$.

To derive the estimator of $\text{Var}({}_{n_1}^{n_2}\hat{\mu}_J)$ we need to use the moment approximation method to obtain approximations for the moments of the statistic ${}_{n_1}^{n_2}\hat{\mu}_J$. The idea is the same as in the case of simple mean estimation. That is, the loss function is expanded with respect to its first argument and evaluated at the point $E(Y_i|X_i) = x_i^T \beta_0$, where β_0 is the true parameter value. In other words, as before, the expansion is evaluated at the true mean.

We list now the assumptions under which our theory holds.

Assumption 1. If S_j is a training set with n_1 number of elements

$$\lim_{n_1 \rightarrow \infty} \frac{1}{n_1} (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} = V$$

where V is finite and positive definite.

Assumption 2. Let x_{n_1k} denote the k th row of the design matrix \mathbf{X}_{S_j} . Then, for each $j = 1, 2, \dots, J$,

$$\max_{1 \leq k \leq n_1} x_{n_1k} (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_{n_1k} \rightarrow 0$$

as $n_1 \rightarrow \infty$.

Notice that this condition is known as the generalized Noether condition.

Under the above conditions $\sqrt{n_1}(\hat{\beta}_{S_j} - \beta)$ converges in distribution to a $N(0, \sigma^2 V)$ random variable.

The following proposition establishes an approximation to the expectation of the loss function L .

Proposition 3.4: Suppose that assumptions 1 and 2 hold. Then

$$E[L(\hat{y}_{i,S_j}, y_i)] = E[L(x_i^T \beta_0, y_i)] + \frac{\sigma^2}{2} E[L''(x_i^T \beta_0, y_i)] \text{tr}[(x_i x_i^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] + R_n,$$

where the remainder term is of order $O(\frac{1}{n_1^2})$, and the prime indicates derivative with respect to the first argument of the loss function.

Proof: First expand $L(\hat{y}_{i,S_j}, y_i)$ with respect to the first argument to obtain:

$$\begin{aligned} L(\hat{y}_{i,S_j}, y_i) &= L(x_i^T \beta_0, y_i) + L'(x_i^T \beta_0, y_i) x_i^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ \frac{1}{2} L''(x_i^T \beta_0, y_i) (\hat{\beta}_{S_j} - \beta_0)^T x_i x_i^T (\hat{\beta}_{S_j} - \beta_0) + R_n, \end{aligned} \quad (3.13)$$

where R_n indicates the remainder term.

Now

$$\begin{aligned} E\{L(\hat{y}_{i,S_j}, y_i)\} &= E_{S_j, i} \{E_{Z_1^n} [L(\hat{y}_{i,S_j}, y_i) | S_j, i]\} \\ &= E_{S_j, i} \{E_{Z_1^n} [L(x_i^T \beta_0, y_i) | S_j, i]\} + E_{S_j, i} \{E_{Z_1^n} [L'(x_i^T \beta_0, y_i) x_i^T | S_j, i] E_{Z_1^n} [(\hat{\beta}_{S_j} - \beta_0) | S_j, i]\} \\ &+ \frac{1}{2} E_{S_j, i} \{E_{Z_1^n} [L''(x_i^T \beta_0, y_i) | S_j, i] E_{Z_1^n} [(\hat{\beta}_{S_j} - \beta_0)^T x_i x_i^T (\hat{\beta}_{S_j} - \beta_0) | S_j, i]\} \end{aligned}$$

But the expectation $E_{Z_1^n} [(\hat{\beta}_{S_j} - \beta_0) | S_j, i] = 0$ because $E_{Z_1^n} (\hat{\beta}_{S_j} | S_j, i) = E_{Z_1^n} (\hat{\beta}_{S_j}) = \beta_0$. Also since the distribution of $\hat{\beta}_{S_j}$ is asymptotically $N(\beta_0, \sigma^2 (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1})$, under assumptions 1 and 2 we obtain:

$$\begin{aligned} E_{S_j, i} \{E_{Z_1^n} [(\hat{\beta}_{S_j} - \beta_0)^T x_i x_i^T (\hat{\beta}_{S_j} - \beta_0) | S_j, i]\} &= E_{Z_1^n} [(\hat{\beta}_{S_j} - \beta_0)^T x_i x_i^T (\hat{\beta}_{S_j} - \beta_0)] \\ &= \sigma^2 \text{tr}[(x_i x_i^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}], \end{aligned}$$

where $\sigma^2 = \text{Var}_{Z_1^n}(X_i)$, the variance of the sample, and $\text{tr}(A)$ stands for the trace of the matrix A . Therefore

$$E[L(\hat{y}_{i,S_j}, y_i)] = E[L(x_i^T \beta_0, y_i)] + \frac{\sigma^2}{2} E[L''(x_i^T \beta_0, y_i)] \text{tr}[(x_i x_i^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] + R_n,$$

where the expectations are taken with respect to the distribution of the data. Moreover, R_n is of order $\frac{1}{n_1^2}$.

Proposition 3.5 establishes the approximation for the variance of $L(\hat{y}_{i,S_j}, y_i)$.

Proposition 3.5 Suppose that assumptions 1 and 2 hold. Then $\text{Var}(L(\hat{y}_{i,S_j}, y_i))$ can be approximated as follows:

$$\begin{aligned} \text{Var}\{L(\hat{y}_{i,S_j}, y_i)\} &= \text{Var}[L(x_i^T \beta_0, y_i)] + \sigma^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} \{Cov(L(x_i^T \beta_0, y_i), \\ &L''(x_i^T \beta_0, y_i)) + E[L'(x_i^T \beta_0, y_i)]^2\} + R_n, \end{aligned}$$

where $\sigma^2 = \text{Var}_{Z_1^n}(Y_i | X_i)$ and R_n is the remaining term of order $\frac{1}{n_1^2}$.

Proof: The proof is similar with that of proposition 3.2, in that we apply proposition 3.4 to $L^2(\hat{y}_{i,S_j}, y_i)$ and we use the fact that

$$[L^2(\hat{y}_{i,S_j}, y_i)]'' = 2L(\hat{y}_{i,S_j}, y_i)L''(\hat{y}_{i,S_j}, y_i) + 2[L'(\hat{y}_{i,S_j}, y_i)]^2,$$

where prime indicates derivative with respect to the first argument of the loss function.

Example. To verify the above approximations we use $L(\hat{y}_{i,S_j}, y_i) = (\hat{y}_{i,S_j} - y_i)^2$, the square error loss and the case of simple regression, that is

$$y_i = a + bz_i + \varepsilon_i = x_i^T \beta + \varepsilon_i,$$

where $x_i^T = (1, z_i)$, $\beta^T = (a, b)$ and $(y_i, x_i) \in S_j^c$. The notation \hat{y}_{i,S_j} stands for $x_i^T \hat{\beta}_{S_j}$.

The exact expectation of $L(\hat{y}_{i,S_j}, y_i) = (x_i^T \hat{\beta}_{S_j} - y_i)^2$ is given as:

$$E[L(\hat{y}_{i,S_j}, y_i)] = \sigma^2 + \sigma^2 x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i.$$

The approximate expectation is

$$E[L(\hat{y}_{i,S_j}, y_i)] = \sigma^2 + \sigma^2 \text{tr}(x_i x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}),$$

Because $\text{tr}(x_i x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}) = x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i$, the approximation to the expectation agrees with the exact computation. Similarly we can verify that the approximation of the variance produces the same result as the exact computation. To illustrate further the formulas assume that $y_i \sim N(x_i^T \beta, \sigma^2)$, then the exact calculation gives the variance of $L(\hat{y}_{i,S_j}, y_i)$,

$$\text{Var}(L(\hat{y}_{i,S_j}, y_i)) = 2\sigma^4 + 4\sigma^4 x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i + 2\sigma^4 (x_i (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i)^2.$$

The approximation is given by

$$\text{Var}(L(\hat{y}_{i,S_j}, y_i)) = 2\sigma^4 + 4\sigma^4 x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i + O\left(\frac{1}{n_1^2}\right),$$

that is they agree up to first order terms.

To complete the variance approximation of the estimator $\frac{n_2}{n_1}\hat{\mu}_J$ we need an approximation of the covariance between $L(\hat{y}_{i,S_j}, y_i)$ and $L(\hat{y}_{i',S_{j'}}, y_{i'})$. The following proposition expresses the approximation of $Cov(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_{j'}}, y_{i'}))$.

Proposition 3.5. Suppose that assumptions 1 and 2 hold. Then for $j \neq j'$, $j, j' \in \{1, 2, \dots, J\}$ when $i \neq i'$

$$\begin{aligned} Cov(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_{j'}}, y_{i'})) &= \sigma^2 (E[L'(x_i^T \beta_0, y_i)])^2 x_i^T (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} x_{i'} \\ &+ \frac{\sigma^4}{2} (E[L''(x_i^T \beta_0, y_i)])^2 tr((x_i x_i^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} (x_{i'} x_{i'}^T) \\ &(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}). \end{aligned}$$

When $i = i'$,

$$\begin{aligned} Cov(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_{j'}}, y_{i'})) &= Var(L(x_i^T \beta_0, y_i)) + \frac{\sigma^2}{2} Cov(L(x_i^T \beta_0, y_i), L''(x_i^T \beta_0, y_i)) \\ &(x_i^T (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} x_i + x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i) \\ &+ \sigma^2 (E[L'(x_i^T \beta_0, y_i)])^2 x_i^T (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i \\ &+ \frac{\sigma^4}{2} (E[L''(x_i^T \beta_0, y_i)])^2 tr((x_i x_i^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} (x_i x_i^T) \\ &(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}) \\ &+ \frac{\sigma^4}{4} Var(L''(x_i^T \beta_0, y_i)) x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i x_i^T (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} x_i. \end{aligned}$$

Proposition 3.6. Let S_j be a training set, $j = 1, 2, \dots, J$. Then for $i \neq i'$

$$\begin{aligned} Cov(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_j}, y_{i'})) &= \sigma^2 (E[L'(x_i^T \beta_0, y_i)])^2 tr[(x_{i'} x_{i'}^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\ &+ \frac{\sigma^4}{2} (E[L''(x_i^T \beta_0, y_i)])^2 tr[(x_i x_i^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (x_{i'} x_{i'}^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}]. \end{aligned}$$

The proofs of Proposition 3.5 and Proposition 3.6 can be found in Appendix C.

Remark: If the loss is square error,

$$Cov(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_j}, y_{i'})) = 2\sigma^4 tr[(x_i x_i^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (x_{i'} x_{i'}^T) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}]. \quad (3.14)$$

To estimate relationship (3.14) we only need to estimate σ . We estimate σ by the residual mean square error.

Under square error loss, we have

$$Var(\hat{\mu}_j) = \frac{1}{n_2} \sum_{i=1}^{n_2} \{2\sigma^4 + 4\sigma^4 x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i\} + \frac{1}{n_2^2} \sum_{i \neq i'} 2\sigma^4 (x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_{i'})^2, \quad (3.15)$$

and

$$\begin{aligned}
 \text{Cov}(\hat{\mu}_j, \hat{\mu}_{j'}) &= \frac{1}{n_2^2} \sum_{i \in S_j^c} \sum_{i' \in S_{j'}^c} \{2\sigma^4 \text{tr}\{(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} \\
 &(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}(x_{i'} x_{i'}^T)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}\} \\
 &\frac{1}{n_2^2} \sum_{i \in S_j^c} \sum_{i' \in S_{j'}^c} \{2\sigma^4 + 4\sigma^4 x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} x_{i'} \\
 &+ 2\sigma^4 \text{tr}\{(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}(x_{i'} x_{i'}^T) \\
 &(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}\} \} \quad (3.16)
 \end{aligned}$$

The final estimate is obtained from relation (3.1) where $\text{Var}(\hat{\mu}_j)$ is estimated by using relation (3.15), $\text{Cov}(\hat{\mu}_j, \hat{\mu}_{j'})$ is estimated by using relation (3.16) and replacing σ^2 by an estimator of it. To obtain an estimator of σ^2 , we fit the regression model and obtain \hat{y}_i . Then $\hat{\sigma}^2$ is the sample variance of the errors $\hat{\epsilon}_i = y_i - \hat{y}_i$, that is the residual mean square.

Remark: Note that to derive the results above, we used as the distribution of the data the conditional distribution of Y given X , in effect treating X as fixed. Now, assume that instead of using the conditional distribution as the data distribution, we treat X as random and use the joint distribution of (X, Y) . In this case, the data distribution is

$$f(x, y) = g(y - x^T \beta | x) k(x)$$

where $g(\cdot)$ is the distribution of the errors and $k(\cdot)$ is the distribution of the x s. We can then derive the formulas expressing the expectation, variance and covariance terms that are needed using the joint distribution of (X, Y) . For example, $E(\hat{\beta}) = E_{(X, Y)}[(X^T X)^{-1} X^T Y] = E_X\{E_{Y|X}[(X^T X)^{-1} X^T Y | X]\} = \beta_0$, is still unbiased, and $\text{Var}(\hat{\beta}) = E_X\{\text{Var}_Y(\hat{\beta} | X)\} + \text{Var}_X\{E_Y(\hat{\beta} | X)\} = \sigma^2 E_X[(X^T X)^{-1}]$. Other adjustments that take into account the distribution of X are needed. These mainly concentrate on taking expectations, over X , of terms that are functions of the X s, and can be easily computed from the data by using bootstrap. As an illustration, under square error loss, the formula in proposition 3.4 becomes $E[L(\hat{y}_{i, S_j}, y_i)] = \sigma^2 + \sigma^2 E_X[\text{tr}\{(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}\}]$, where σ^2 is the variance of the error distribution.

4. Simulation Experiments.

We present here simulation experiments that illustrate the performance of the proposed estimators; moreover, we compare these estimators with the estimator proposed by Nadeau and Bengio (2003). The simulation experiments compare the proposed estimators with the Nadeau and Bengio estimator under two different error losses, the square error and the absolute error loss.

4.1 Square Error Loss

We will first describe the experimental setup for the simple mean case.

We generated data sets of size $n = 100$ from a $N(0, 1)$ distribution in S-plus. For each different size n_1 of the training set S_j we randomly select n_1 data points from the available n and use S_j^c , the complement of S_j with respect to the generated data universe that contains 100 data points, as a test

set. We take J to be 15 (as recommended by Nadeau and Bengio, 2003), and 50. We then computed $S_{\hat{\mu}_j}^2 = \frac{1}{J-1} \sum_{j=1}^J (\hat{\mu}_j - \frac{n_2}{n_1} \hat{\mu}_J)^2$ and the estimator of the variance of the generalization error, given as $(\frac{1}{J} + \frac{n_2}{n_1}) S_{\hat{\mu}_j}^2$.

We also computed the moment approximation estimator given by expressions (3.9) and (3.10). Notice that we estimate σ^2 by using the sample variance, that is, $\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2$. We also computed the variance estimator of $\frac{n_2}{n_1} \hat{\mu}_J$ using expression,

$$\frac{1}{J} \frac{1}{n_2} Var(X_i^2) + \frac{1}{J} \frac{4\sigma^4}{n_1 n_2} + \frac{J-1}{J} \left\{ \frac{1}{n} \left(\frac{4\sigma^4}{n} - \frac{\sigma^4}{n_1^2} + Var(X_i^2) \right) - \left(1 - \frac{1}{n}\right) \frac{\sigma^4}{n_1^2} \right\}.$$

The population variance σ^2 is estimated by using the sample variance averaged over 100 different data sets. The term $Var(X_i^2)$ is estimated as follows. Let $Z_i = X_i^2, i = 1, 2, \dots, n$. We created a new data universe using Z_i and estimate $\hat{Var}(Z_i) = \frac{1}{n-1} \sum_{i=1}^n (Z_i - \bar{Z})^2$, where $\bar{Z} = \frac{1}{n} \sum_{i=1}^n Z_i$, over 100 different data sets.

Table 1 presents the results of the simulation. The first column of the table shows the size of the test set. The second column reports the value of the Nadeau and Bengio estimator, while the third column reports its variance. The variance is computed by simply taking the sample variance of the estimator that was computed over the 100 independent data sets. The fourth column of the table reports the value of the moment approximation estimator of the variance of the cross validation estimator of the generalization error, while the fifth column reports the sample variance of the moment approximation estimator.

n_2	NB	var(NB)	MA	var(MA)
10	0.0316	0.000310	0.0328	7.75e-06
15	0.0265	0.000241	0.0282	5.34e-05
20	0.0250	0.000179	0.0259	4.50e-05
25	0.0235	0.000213	0.0245	4.03e-05
30	0.0238	0.000145	0.0236	3.73e-05
35	0.0227	0.000175	0.0229	3.52e-05
40	0.0235	0.000188	0.0224	3.36e-05
45	0.0227	0.000122	0.0219	3.23e-05
50	0.0246	0.000236	0.0216	3.13e-05

Table 1: Simple mean case $n=100, J=15$. Nadeau-Bengio (NB) and moment approximation (MA) estimators of the variance of the cross validation estimator of the generalization error, and their sample variances. $J = 15$, and the results are averages over 100 independent data sets. The size of the data universe is 100.

We notice that the variance of the moment approximation estimator is at least one order of magnitude smaller than the variance of the Nadeau- Bengio estimator, thereby increasing the accuracy of the moment estimator.

Figure 1 plots the values of the Nadeau-Bengio and moment approximation estimate of the variance versus the sample size of the test set. Notice that the curve corresponding to the moment approximation is smooth. This is in contrast to the behavior of the Nadeau-Bengio estimator, which

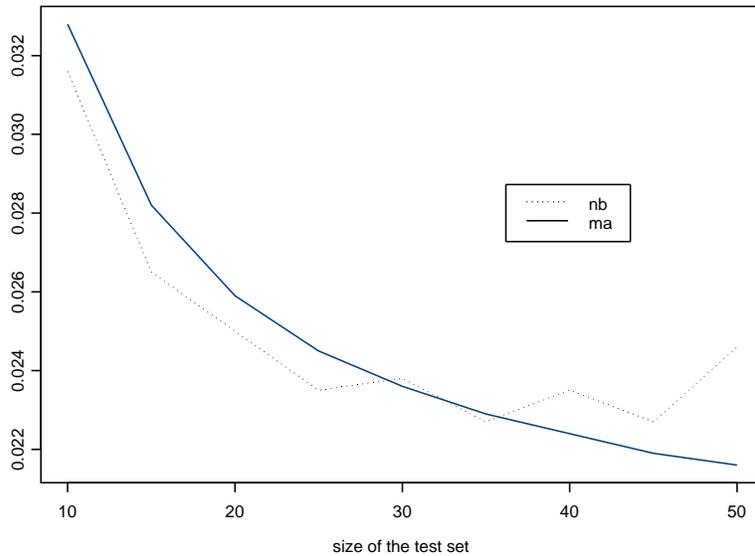


Figure 1: Simple mean case $n=100, J=15$

seems to fluctuate (this also is indicated by the value of the sample variance associated with the estimator and reported in table 1.)

n_2	NB	var(NB)	MA	var(MA)
10	0.0235	1.24e-04	0.0241	7.75e-06
15	0.0212	8.77e-05	0.0227	3.47e-05
20	0.0211	6.27e-05	0.0220	3.26e-05
25	0.0204	7.50e-05	0.0216	3.13e-05
30	0.0206	7.28e-05	0.0213	3.05e-05
35	0.0203	6.79e-05	0.0211	2.98e-05
40	0.0204	7.94e-05	0.0209	2.93e-05
45	0.0213	8.08e-05	0.0207	2.88e-05
50	0.0206	6.43e-05	0.0206	2.84e-05

Table 2: Simple mean case $n=100, J=50$. Moment approximation (MA) and Nadeau-Bengio (NB) estimators of the variance the cross validation estimator of the generalization error and their sample variances. $J = 50$, and the results are averages over 100 independent data sets. The size of the data universe is 100.

Table 2 presents the variance estimates of the CV estimators of the generalization error when $J = 50$. In this case we notice that the variance of the moment approximation estimator is about half of the variance of the Nadeau-Bengio estimator.

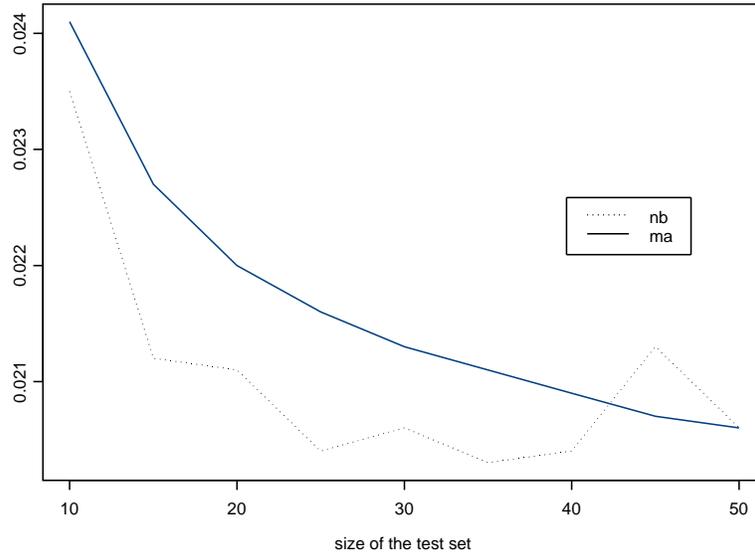


Figure 2: Simple mean case $n=100, J=50$

Figure 2 shows a plot of Nadeau-Bengio and moment approximation estimate of the variance as a function of the size of the test set. The larger variance of the Nadeau-Bengio estimator that was reported in table 2 can also be seen again in Figure 2.

Table 3 presents the values of the two variance estimators as well as their variance when the data universe has size $n = 1000$, for the case $J = 15$ and $J = 50$. We notice that the performance, in terms of variance, of the moment approximation estimator is, in both cases, superior to the performance of the Nadeau-Bengio estimator, always having variance that is smaller than the NB variance by one order of magnitude.

To address the problem of bias we computed the exact (and theoretical) value of the variance estimator of $\frac{n_2}{n_1} \hat{\mu}_J$. Therefore, we computed, using formula (3.1), $Var(\frac{n_2}{n_1} \hat{\mu}_J)$ under square error loss and under the assumption of a $N(0, 1)$ distribution. The distributional assumption is used to obtain the theoretical value. This is done only for the purpose of comparison and in order to allow a bias computation to be carried out without having to estimate higher order moments. In practice, the distribution of the population from which the data arise is not known, and higher order moments need to be estimated from the data.

The exact theoretical value of $Var(\hat{\mu}_j)$ is

$$Var(\hat{\mu}_j) = \frac{2}{n_2} \left\{ 1 + \frac{2}{n_1} + \frac{n_2}{n_1^2} \right\}.$$

n_2	NB	var(NB)	MA	var(MA)
J=15				
100	0.00319	1.61e-06	0.00319	7.75e-06
150	0.00291	1.22e-06	0.00275	5.42e-08
200	0.00252	9.62e-07	0.00253	4.58e-08
250	0.00244	8.21e-07	0.00239	4.11e-08
300	0.00240	9.02e-07	0.00230	3.81e-08
350	0.00214	9.27e-07	0.00224	3.60e-08
400	0.00232	7.21e-07	0.00219	3.45e-08
450	0.00217	5.70e-07	0.00216	3.33e-08
500	0.00206	8.24e-07	0.00213	3.24e-08
J=50				
100	0.00241	3.20e-07	0.00235	5.90e-08
150	0.00225	2.82e-07	0.00222	3.54e-08
200	0.00225	3.68e-07	0.00215	3.33e-08
250	0.00216	2.43e-07	0.00211	3.21e-08
300	0.00213	1.96e-07	0.00209	3.12e-08
350	0.00216	2.83e-07	0.00207	3.07e-08
400	0.00211	2.70e-07	0.00205	3.02e-08
450	0.00218	2.36e-07	0.00204	2.99e-08
500	0.00206	2.18e-07	0.00203	2.96e-08

Table 3: Simple mean case $n=1000$, $J=15$ and $J=50$. Moment approximation (MA) and Nadeau-Bengio (NB) estimators of the variance of the cross validation estimator of the generalization error under random selection, and their sample variances. The size of the data universe is $n = 1000$ and $J = 15$ and 50 .

Using theorem 3.1 the approximation to the value of $Var(\hat{\mu}_j)$ is

$$Var(\hat{\mu}_j) = \frac{2}{n_2} \left\{ 1 + \frac{2}{n_1} + O\left(\frac{1}{n_1^2}\right) \right\}.$$

The same theorem provides the approximation to $Cov(\hat{\mu}_j, \hat{\mu}_{j'})$ as follows:

$$Cov(\hat{\mu}_j, \hat{\mu}_{j'}) = \frac{2}{n} \left(1 + \frac{2}{n} \right) + O\left(\frac{1}{n_1^2}\right).$$

The exact theoretical computation of the covariance provides us with the formula

$$Cov(\hat{\mu}_j, \hat{\mu}_{j'}) = \frac{2}{n} \left(1 + \frac{2}{n} \right) + \frac{2}{n_1} \left(\frac{1}{n_1} - \frac{1}{n} \right).$$

Using these expressions we computed the exact value of the variance of $\frac{n_2}{n_1} \hat{\mu}_J$ for the square error loss. This computation allows us to get a sense of the bias of the moment approximation and Nadeau-Bengio estimators. Table 4 presents the results for the case where the data universe is 100

n_2	Exact Variance	Bias of MA estimator	Bias of NB estimator
10	0.0327	0.0001	-0.0011
15	0.0282	0	-0.0017
20	0.0259	0	-0.0009
25	0.0246	-0.0001	-0.0011
30	0.0237	-0.0001	0.0001
35	0.0232	-0.0003	-0.0005
40	0.0227	-0.0003	0.0008
45	0.0223	-0.0004	0.0004
50	0.0222	-0.0006	0.0024

Table 4: Bias of MA and NB estimators. Bias of MA of NB estimators for the case of the simple mean. The data universe has size 100, $J=15$. The bias is calculated as the expectation of the estimator minus the exact value.

and $J = 15$. We observe that the moment approximation estimator has a very small bias, consistently smaller than the bias of the Nadeau-Bengio estimator. Notice that when the sizes of the training and test sets are equal ($n_1 = n_2 = 50$) the bias of the Nadeau-Bengio estimator is four times higher, in absolute value, than that of the moment approximation estimator.

At this point, we remind the reader that the Nadeau-Bengio estimator given in (2.5) is generally applicable. The proposed estimators take advantage of information about the data and the learning algorithm. Hence, it is not completely surprising that they perform better than the Nadeau Bengio estimator in terms of variance and bias.

For comparison reasons, after a referee’s suggestion, we computed the second estimator proposed by Nadeau and Bengio(2003) and given by (2.6). Table 5 presents the values of the estimators of the variance given by (2.5) and (2.6) and the moment approximation estimator. Expressions (3.9) and (3.10) were used to obtain the needed variance and covariance terms. The size of the data universe is 50, 100, 500 and 1000, the size of the test set is taken to be 10, 20, 100 and 200 and J is either 15 or 50. From table 5 we see that the estimator given by (2.6) is indeed conservative; its value is almost twice as big as the value of either the cheap to compute Nadeau and Bengio estimator given by (2.5) and the moment approximation estimator. It is interesting to notice that, when the training set size is the same with the training set size used to compute (2.5) and the moment approximation estimator, the value of (2.6) is comparable to the value of the other two estimators. This observation indicates the importance of the size of the training set in the computation of the variance of the cross-validation estimators of the generalization error.

To exemplify the fact that the framework we propose allows one to compute the variance estimator of the k-fold cross validation estimator of the generalization error we computed the variance of leave-one-out cross validation (LOOCV) estimator of the generalization error, the 4-fold, the 5-fold and the 10-fold in the case of square error loss and when the data universe consisted of 100 data points generated from a $N(0,1)$ distribution. The case was prediction of simple mean. We did the same when the data universe consisted of 1000 normal data points. Table 6 presents the moment approximation variance estimators together with their variance and the corresponding NB estimators.

Sample Size	Training Set Size	J	NB	MA	NB(Conserv.)
50	10	15	0.0539	0.0537	0.0988
100	10	15	0.0314	0.0328	0.0542
50	20	15	0.0458	0.0462	0.1213
100	20	15	0.0257	0.0259	0.0456
50	10	50	0.0443	0.0456	0.0836
100	10	50	0.0236	0.0241	0.0420
50	20	50	0.0421	0.0430	0.1131
100	20	50	0.0218	0.0220	0.0467
500	100	15	0.0052	0.0051	0.0081
1000	100	15	0.0032	0.0032	0.0050
500	200	15	0.0044	0.0044	0.0082
1000	200	15	0.0025	0.0025	0.0041
500	100	50	0.0044	0.0043	0.0078
1000	100	50	0.0023	0.0023	0.0040
500	200	50	0.0042	0.0041	0.0081
1000	200	50	0.0022	0.0022	0.0040

Table 5: Comparison among three estimators. Values of NB, MA and the conservative NB estimates for the case of the simple mean. The universe sample size is 50, 100, 500 and 1000.

	k-fold	MA	Variance	NB	Variance
n=100	4-fold	0.02096	0.00003302	0.0417	0.001262
	5-fold	0.02093	0.00003293	0.04516	0.0009909
	10-fold	0.02089	0.0000328	0.04426	0.0005567
	LOOCV	0.02086	0.0000327	0.04141	0.0002177
n=1000	4-fold	0.002	3.02E-08	0.00423	1.308E-05
	5-fold	0.002	3.02E-08	0.00412	8.60E-06
	10-fold	0.002	3.02E-08	0.00405	3.74E-06
	LOOCV	0.002	3.02E-08	0.00398	2.00E-07

Table 6: Variance estimators for k-fold CV. Moment approximation and Nadeau-Bengio variance estimators for k-fold cross-validation estimators of the generalization error and their variances.

When the data universe is 100 the 4-fold cross validation divides it into 4 non-overlapping test sets each containing 25 data points. Similarly, we define 5-fold and 10-fold cases. We notice that the variance estimation of LOOCV is not appreciably better than that of the other cross validation estimators. In fact, the slight advantage of the LOOCV diminishes when the data universe is large and the size of the test set becomes large. For illustration purposes we present the NB estimator and its variance. The value of the NB estimator is twice as large as the value of the moment

approximation estimator. However, note that Nadeau and Bengio (2003) do not discuss the case of k-fold cross validation.

K	MSE	Var	Bias
4	0.02123	0.02098	0.002283
10	0.02099	0.02091	0.002224

Table 7: Comparison between 10-fold and 4-fold Cross Validation Under Simple Mean Case. MA estimator is used to estimate the variance of the cross -validation estimator of the generalization error. The results reported in the table are averages over 100 different data sets.

To understand the effect of the loss function in the performance of the methods we used the mean squared error (MSE) to compare the estimators as well as their variance. Table 7 presents the values of the MSE and the variance, as well as the bias for the 4-fold and 10-fold estimators of variance for the simple mean case. We see that the reduction in variance between the 4-fold and 10-fold CV variance estimator is not appreciably different. This difference is more pronounced when the corresponding MSE are compared. Overall it appears that the 10-fold cross validation differs from the 4-fold cross validation an order of magnitude less when the comparison between the two is made on the basis of variance than when the comparison is made on the basis of MSE.

4.2 Absolute Error Loss

The previous theory was developed for loss functions that are differentiable. One loss that is not differentiable at the mean is the absolute error loss. However, we are able to apply the above theory in the case of the absolute error loss because we can replace $|\bar{X}_{S_j} - X_i|$ by the equivalent function $\sqrt{(\bar{X}_{S_j} - X_i)^2 + d}$, where d is a small positive number. The function $[(\bar{X}_{S_j} - X_i)^2 + d]^{1/2}$ replaces the absolute error loss and is differentiable everywhere. We use $d = \frac{1}{n}$ and $\frac{1}{n_2}$ and computed the Nadeau-Bengio estimate and the moment approximation estimate for the sizes of the data universe of 100 and 500. Notice that the Nadeau-Bengio estimate was computed using $L(\bar{X}_{S_j}, X_i) = |X_i - \bar{X}_{S_j}|$, while the moment approximation estimator uses the loss function $L(\bar{X}_{S_j}, X_i) = [(\bar{X}_{S_j} - X_i)^2 + d]^{1/2}$, which is almost the same with the absolute error loss. We generate data from a $N(0, 5)$ distribution in S-plus and used $J = 15$.

Table 8 shows the values of the Nadeau-Bengio and moment approximation estimators together with their sample variances. Notice that $d = \frac{1}{n}$ was used in the first computation of the moment approximation estimator, where n is the size of the data universe, and $d = \frac{1}{n_2}$, where n_2 is the size of the test set was used in the second computation. The table reports results that are averaged over 100 different data sets.

The first observation we make is that the effect of d on the moment approximation estimator and its sample variance is almost undetectable, as the values of the estimator and its sample variance (averaged over 100 different data sets) do not change with d being $\frac{1}{n}$ or $\frac{1}{n_2}$. Secondly, we see that the variance of the Nadeau-Bengio estimator is larger than the variance of the moment approximation estimator by one order of magnitude.

n_2	NB estimator	var(NB)	MA estimator	var(MA)
$d = \frac{1}{n}$				
10	0.0287	1.16e-04	0.0293	1.43e-05
15	0.0271	1.25e-04	0.0252	1.06e-05
20	0.0256	7.93e-05	0.0231	8.93e-06
25	0.0224	7.72e-05	0.0219	7.98e-06
30	0.0218	8.77e-05	0.0210	7.36e-06
35	0.0207	7.53e-05	0.0204	6.92e-06
40	0.0208	6.52e-05	0.0199	6.58e-06
45	0.0191	6.14e-05	0.0194	6.30e-06
50	0.0205	7.07e-05	0.0191	6.06e-06
$d = \frac{1}{n_2}$				
10	0.0287	1.16e-04	0.0291	1.43e-05
15	0.0271	1.25e-04	0.0251	1.06e-05
20	0.0256	7.93e-05	0.0231	8.92e-06
25	0.0224	7.72e-05	0.0218	7.97e-06
30	0.0218	8.77e-05	0.0210	7.36e-06
35	0.0207	7.53e-05	0.0204	6.91e-06
40	0.0208	6.52e-05	0.0199	6.58e-06
45	0.0191	6.14e-05	0.0194	6.30e-06
50	0.0205	7.07e-05	0.0191	6.06e-06

Table 8: Absolute Error Loss Case $n=100$, $J=15$. Nadeau-Bengio (NB) and moment approximation (MA) estimators and their corresponding variance estimates. Data are $N(0,5)$ and $J=15$. The loss function is absolute error.

Table 9 presents the Nadeau-Bengio and moment approximation estimators but now the value of $J = 50$. Notice that, in contrast with the square error loss case, the Nadeau-Bengio estimator has a higher variance than the moment approximation estimator. Its variance is still an order of magnitude higher than the variance of the moment approximation estimator.

Table 10 presents the two estimators and their corresponding sample variances when the size of the data universe is 500. The population is still $N(0,5)$ and $d = 1/n$. Notice that for $J = 15$ the NB estimate has larger, by two orders of magnitude, variance than the moment approximation estimator, while $J = 50$ it still maintains a larger than the moment approximation estimator variance, only this time by one order of magnitude.

4.3 Regression

In the regression case the data generation was done as follows. The model adopted was simple regression, that is $y_i = \alpha + \beta x_i + \varepsilon_i$, $i = 1, 2, \dots, n$, where ε_i are independent, mean 0 and variance 1, normal random variables. The parameters α , β were set to equal 2 and 3 respectively. The explanatory variable was generated from a uniform distribution with range $[0,10]$. Finally, we generated the errors from a $N(0,1)$ distribution and $y_i = 2 + 3x_i + \varepsilon_i$, $i = 1, 2, \dots, 100$. We generated 100 different

n_2	NB	var(NB)	MA	var(MA)
$d = \frac{1}{n}$				
10	0.0208	3.18e-05	0.0216	7.77e-06
15	0.0209	2.48e-05	0.0203	6.89e-06
20	0.0206	2.92e-05	0.0197	6.46e-06
25	0.0189	2.30e-05	0.0193	6.19e-06
30	0.0199	2.41e-05	0.0190	6.00e-06
35	0.0191	2.45e-05	0.0187	5.86e-06
40	0.0192	2.49e-05	0.0185	5.73e-06
45	0.0188	3.16e-05	0.0184	5.61e-06
50	0.0195	2.56e-05	0.0182	5.50e-06
$d = \frac{1}{n_2}$				
10	0.0208	3.18e-05	0.0214	7.75e-06
15	0.0209	2.48e-05	0.0202	6.88e-06
20	0.0206	2.92e-05	0.0196	6.45e-06
25	0.0189	2.30e-05	0.0192	6.19e-06
30	0.0199	2.41e-05	0.0190	6.00e-06
35	0.0191	2.45e-05	0.0187	5.86e-06
40	0.0192	2.49e-05	0.0185	5.73e-06
45	0.0188	3.16e-05	0.0183	5.61e-06
50	0.0195	2.56e-05	0.0182	5.50e-06

Table 9: Absolute Error Loss Case $n=100$, $J=50$. Nadeau-Bengio (NB) and moment approximation (MA) estimators and their sample variance. Data are $N(0,5)$ and $J=50$. The loss function is absolute error.

data sets; for each data set, and for each value of n_2 , n_1 we computed the Nadeau-Bengio and the moment approximation estimator and then average those over the 100 different data sets.

Tables 11 and 12 present the two estimators together with their corresponding sample variances and for values of J equal to 15 and 50. Notice that the moment approximation estimator has variance that is at least one order of magnitude smaller than the variance of Nadeau-Bengio estimator.

Table 13 computes the NB and moment approximation variance estimators of the generalization error when the size of the data universe is 500. We see that the moment approximation estimator still maintains a variance of an order of magnitude lower than the NB estimator.

We also computed the variance estimators for k-fold cross validation estimators of the generalization error in the regression case. Table 14 shows the value of the moment approximation and Nadeau-Bengio estimator and their sample variances computed over 100 different data sets of size 100.

Again, the advantage of LOOCV in this case is questionable. Moreover, given the fact that 4-fold cross validation saves a lot of computing time it seems to be preferable to use (recall that 4-fold CV assigns 25% of the data points in the test set).

n_2	NB	var(NB)	MA	var(MA)
$J = 15$				
50	0.00651	8.31e-06	0.00588	1.09e-07
75	0.00527	3.19e-06	0.00506	8.04e-08
100	0.00455	3.23e-06	0.00465	6.79e-08
125	0.00459	2.62e-06	0.00440	6.09e-08
150	0.00428	3.10e-06	0.00424	5.64e-08
175	0.00420	2.55e-06	0.00412	5.33e-08
200	0.003971	2.41e-06	0.00403	5.10e-08
225	0.00390	1.83e-06	0.00396	4.92e-08
250	0.00361	2.03e-06	0.00390	4.78e-08
$J = 50$				
50	0.00456	1.05e-06	0.00433	5.90e-08
75	0.00402	6.61e-07	0.00409	5.25e-08
100	0.00406	9.03e-07	0.00396	4.93e-08
125	0.00404	7.45e-07	0.00389	4.75e-08
150	0.00396	7.16e-07	0.00384	4.62e-08
175	0.00388	8.07e-07	0.00380	4.54e-08
200	0.00377	5.23e-07	0.00377	4.47e-08
225	0.00377	5.67e-07	0.00375	4.41e-08
250	0.00365	6.26e-07	0.00373	4.36e-08

Table 10: Absolute Error Loss Case $n=500$, $d = \frac{1}{n}$. Nadeau-Bengio (NB) and moment approximation (MA) estimators and their sample variance. The size of the data universe is 500.

4.4 Classification

In this section we briefly indicate how these results can possibly be extended to the classification case. We present some ideas that appear promising in treating this case and a very limited simulation experiment in the simplest case, where the prediction rule is based on the mean of the training set. The results presented here are promising; however, we would like to stress that a more detailed study than the one presented here, is required to understand the performance of these methods in classification.

Recall that a central requirement on the loss function is to be differentiable. In the classification case the loss function is an indicator function and hence it is discontinuous at one point. The idea is to replace the discontinuous function by a continuous, differentiable function that is close to the original loss function. We approximate therefore the indicator function by a polynomial of order 3. Let the data be (x_i, g_i) , $i = 1, \dots, n$, where x_i indicates the data value, and g_i indicates the group membership. Assume that there are only two groups in the population; then $g_i = 1$ if x_i belongs in group 1 and $g_i = 2$ if x_i belongs in group 2. Moreover, assume that group 1 has smaller mean than group 2. The prediction rule we use states that if $\bar{X}_{S_j} - X_k > 0$ then X_k belongs in group 1, otherwise it belongs in group 2. Therefore, \hat{g}_k is either 1 or 2 depending on whether $\bar{X}_{S_j} - X_k$ is greater than 0 or less than or equal to 0. The loss function is then $I(g_k \neq \hat{g}_k)$.

n_2	NB	var(NB)	MA	var(MA)
10	0.0327	0.000493	0.0326	1.14e-04
15	0.0293	0.000366	0.0284	8.44e-05
20	0.0259	0.000184	0.0260	7.21e-05
25	0.0242	0.000199	0.0247	6.29e-05
30	0.0235	0.000168	0.0238	5.74e-05
35	0.0226	0.000176	0.0232	5.66e-05
40	0.0235	0.000144	0.0227	5.35e-05
45	0.0249	0.000255	0.0223	5.16e-05
50	0.0233	0.000142	0.0221	5.06e-05

Table 11: Regression case $n=100$, $J=15$. Moment approximation (MA) and Nadeau-Bengio (NB) estimators of the variance of the cross validation estimator of the generalization error and their sample variances in the regression case. The value of J is 15, and the results are averages over 100 independent data sets. The size of the data universe is 100.

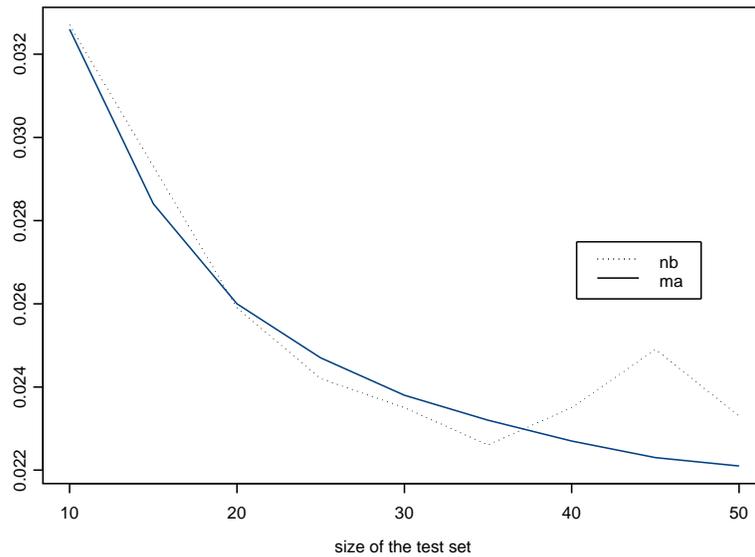


Figure 3: Regression case $n=100$, $J=15$

We can write this loss function as a function of $z_k = \bar{x}_{S_j} - x_k$, $\delta_k = I(g_k = 1)$ and two continuous differentiable functions L_{k1} and L_{k2} . Thus

$$I(g_k \neq \hat{g}_k) = \delta_k L_{k1} + (1 - \delta_k) L_{k2},$$

n_2	NB	var(NB)	MA	var(MA)
10	0.0253	1.84e-04	0.0242	6.00e-05
15	0.0233	1.29e-04	0.0229	5.41e-05
20	0.0228	1.24e-04	0.0222	5.06e-05
25	0.0223	1.15e-04	0.0218	4.92e-05
30	0.0219	1.07e-04	0.0215	4.79e-05
35	0.0222	1.10e-04	0.0213	4.70e-05
40	0.0215	1.00e-04	0.0212	4.63e-05
45	0.0231	1.31e-04	0.0211	4.60e-05
50	0.0231	9.56e-05	0.0210	4.54e-05

Table 12: Regression case $n=100$, $J=50$. Moment approximation (MA) and Nadeau-Bengio (NB) estimators of the variance of the cross validation estimator of the generalization error and their sample variances in the regression case. The value of J is 50, and the results are averages over 100 independent data sets. The size of the data universe is 100.

n_2	NB	var(NB)	MA	var(MA)
50	0.00653	7.64e-06	0.00643	8.94e-07
75	0.00563	4.80e-06	0.00555	6.71e-07
100	0.00498	4.10e-06	0.00511	5.92e-07
125	0.00470	3.86e-06	0.00483	5.02e-07
150	0.00495	4.35e-06	0.00464	4.54e-07
175	0.00469	3.57e-06	0.00452	4.32e-07
200	0.00450	2.42e-06	0.00443	4.16e-07

Table 13: Regression case $n=500$, $J=15$. Moment approximation (MA) and Nadeau-Bengio (NB) estimators of the variance of the cross validation estimator of the generalization error and their sample variances in the regression case. The value of J is 15, and the results are averages over 100 independent data sets. The size of the data universe is 500.

where

$$L_{k1} = \begin{cases} 1 & , z_k < -h \\ \frac{2}{h^3} z_k^3 + \frac{3}{h^2} z_k^2 & , -h \leq z_k < 0 \\ 0 & , z_k \geq 0 \end{cases}$$

$$L_{k2} = \begin{cases} 0 & , z_k < 0 \\ -\frac{2}{h^3} z_k^3 + \frac{3}{h^2} z_k^2 & , 0 \leq z_k < h. \\ 1 & , z_k \geq h \end{cases}$$

The needed terms then can be easily computed. For example, we can compute expectation of the above loss function as

$$E\{E(\delta_k L_{k1} + (1 - \delta_k) L_{k2} | \delta_k)\} = P(\delta_k = 1)E(L_{k1} | \delta_k = 1) + P(\delta_k = 0)E(L_{k2} | \delta_k = 0)$$

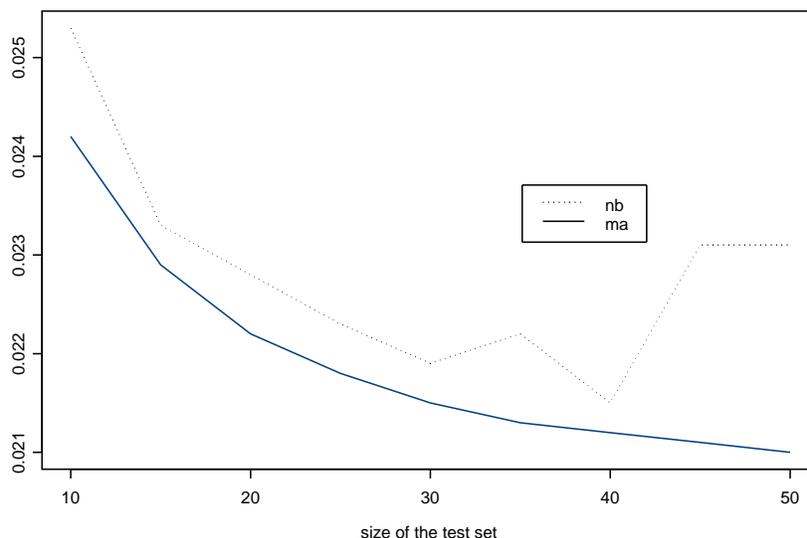


Figure 4: Regression case n=100, J=50

k-fold	MA	Variance	NB	Variance
4-fold	0.02132	0.0000357	0.04854	0.00227
5-fold	0.02135	0.0000358	0.04634	0.00121
10-fold	0.02138	0.0000359	0.04493	0.00062
LOOCV	0.02139	0.0000359	0.04323	0.00023

Table 14: Variance estimators in regression. Variance estimators of k-fold cross-validation estimator of the generalization error and their sample variances, in regression.

and the terms $P(\delta_k = 1)$, $P(\delta_k = 0)$ are computed from the data. Similarly, we can compute from the data all terms that involve variance and covariance terms.

Table 15 presents the results obtained from a small scale simulation. Data were generated in *Splus* from two groups of normal distributions; these were $N(3, 1)$ and $N(1, 1)$. Group membership is assigned by generating a Bernoulli(0.6) random variable. If the value of 1 is obtained then the data point is generated from a $N(1, 1)$ distribution, otherwise it is generated from a $N(3, 1)$. The training set used 80% of the available data points. For example, when $n = 200$ the training set contains 160 elements and thus $n_2 = 40$. The value of h in constructing the L_{k1} , L_{k2} functions was taken to be 0.1.

Table 15 shows the moment approximation variance estimator and NB estimator for various values of the data universe. For illustration reasons we present the values of the MA estimator for both cases when normality is assumed and when is not. We see that the moment approximation estimator (computed without any distributional assumption) is very competitive.

Table 15: Simple Classification Example

n	MA.Free	MA.Normal	NB
200	0.0008355	0.0008275	0.0009240
2000	0.00008593	0.00008273	0.00010028
20000	0.000008603	0.000008299	0.000008815

Table 15. Moment approximation (MA) and Nadeau-Bengio (NB) estimators of the variance of the cross validation estimator of the generalization error and their sample variances in the simple classification case. The value of J is 15, MA.Free denote the MA estimator without distribution assumption and MA.Normal denote the MA estimator under normal distribution. The results are averages over 100 independent data sets. 80 percent of the data are used as training data; h used here is 0.1

5. Discussion and Conclusion

We presented a method for deriving variance estimators of the cross validation estimator of the generalization error in the cases of smooth loss functions and the absolute error loss. The approximation we propose illustrates clearly the role of the training and test sets in the estimation of the variance of the generalization error. We also provide a unifying framework, under which we can obtain variance estimators of the estimators of the generalization error for both, complete random sampling and non-random test set selection.

We compared the moment approximation estimators with an estimator proposed by Nadeau and Bengio (2003). The results indicate that the moment approximation estimators perform better in terms of both, variance and bias, than the Nadeau and Bengio (2003) estimator. The new estimators use additional information from both the data and the learning algorithm. On the other hand, the Nadeau and Bengio estimator is computationally simpler than the moment approximation estimator for general loss functions, as it does not require the computation of the derivatives of the loss function. In the case of non-random test set selection, the Nadeau-Bengio estimator is not appropriate to use. The moment approximation estimator in this case is a reasonable estimator and can be computed. It is interesting to notice that the results indicate against use of the leave-one-out cross validation (LOOCV). Its slight advantage in terms of variance, over the other forms of cross-validation quickly diminishes as the size of the universe, and hence the size of the test set of other cross validation schemes increases. Overall, a test set that use 25% of the available data seems to be a reasonable compromise in selecting among the various forms of k-fold cross validation.

We presented results for general differential loss functions and for absolute error loss. We also indicated possible extensions of this methodology to the classification problem and discussed briefly a very simple version of the classification problem. An extensive study of this problem will be the subject of a different paper. Finally, we would like to indicate here that the methods presented here can similarly apply to SVM loss function as well as the kernel regression.

Acknowledgments

The first author would like to acknowledge support for this project from the National Science Foundation (NSF grants DMS-0072319 and DMS-0504957). The last author would like to acknowledge

the support from the National Library of Medicine (R01-LM08910), NIH. The authors would like to thank two referees for their constructive suggestions that improve the presentation of the paper.

Appendix A.

Here we present a series of lemmas that guarantee that the remainder term in the approximations for the case of sample mean.

Before we state these we need the following definitions.

Definition 1. Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space. We say that a random variable X belongs in the \mathcal{L}_p space if $E|X|^p < \infty, p > 0$.

Definition 2. A sequence of random numbers R_n is said to be $O(1/k_n)$ if $\exists M$ and n_0 such that $|k_n R_n| < M, \forall n > n_0$, or, equivalently, $k_n R_n$ is bounded.

Lemma A.1 Let X, Y be independent random variables and $X + Y \in \mathcal{L}_r$ for some $r \in (0, \infty)$. Then $X \in \mathcal{L}_r$ and $Y \in \mathcal{L}_r$.

Proof. For a large $\lambda_0 > 0, \forall \lambda > \lambda_0$

$$\begin{aligned} P(|X| > \lambda) &\leq 2P(|X| > \lambda, |Y| < \frac{\lambda}{2}) \\ &\leq 2P(|X + Y| > \frac{\lambda}{2}), \end{aligned}$$

If $E|X|^r < +\infty$, then $E|X|^r = \int_0^\infty P[|X|^r > \lambda]d\lambda$. Hence, if $X + Y \in \mathcal{L}_r$,

$$\begin{aligned} \int_{\lambda \geq \lambda_0} P(|X|^r > \lambda) &= \int_{\lambda \geq \lambda_0} P(|X| > \lambda^{\frac{1}{r}})d\lambda \\ &\leq 2 \int_{\lambda \geq \lambda_0} P(|X + Y| > \frac{\lambda^{\frac{1}{r}}}{2})d\lambda \\ &= 2 \int_{\lambda \geq \lambda_0} P(|X + Y|^r > \frac{\lambda}{2^r})d\lambda < \infty. \end{aligned}$$

Thus, $E|X|^r < \infty$. The proof for $E|Y|^r < \infty$ is similar.

Lemma A.2 If $0 < r' < r$ and $E|X|^r < \infty$, then $E|X|^{r'} < \infty$.

Proof. Write

$$(E|X|^{r'})^{r/r'} \leq E(|X|^{r'})^{r/r'} = E|X|^r < \infty,$$

and the proof is obtained by Jensen's inequality.

Lemma A.3 If $E|\bar{X}_n|^p < \infty$, then $E|X_1|^p < +\infty$, where $p \in \mathcal{Z}^+$, and $\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$ is the sample mean.

Proof. We will use transfinite induction. For $n = 1$ and $n = 2$, it is trivial since $\bar{X}_n = X_1$. For $n = 2, \bar{X}_2 = \frac{1}{2}(X_1 + X_2)$ and use lemma 1 to obtain the result, relying on the fact that X_1, X_2 are identically distributed. Suppose now that for $n \leq k - 1$ the result holds. We will prove it true for $n = k$. Write

$$E(|\bar{X}_k|^p) = E(|\frac{1}{k} \sum_{i=1}^k X_i|^p) = E(|\frac{1}{k} (\sum_{i=1}^{k-1} X_i + X_k)|^p).$$

Thus

$$E|\bar{X}_k|^p = E\left|\frac{1}{k}X_k + \frac{k-1}{k}\bar{X}_{k-1}\right|^p,$$

and using lemma 1, we obtain $E(|X_k|) < \infty$.

Lemma A.4 Let $n > 2k$ and a_1, a_2, \dots, a_n be such that

$$\left. \begin{array}{l} a_1 + a_2 + \dots + a_n = 2k \\ a_i \in \mathbb{Z}, a_i \geq 0, a_i \neq 1 \end{array} \right\} \quad (1)$$

$$\left. \begin{array}{l} a_1 + a_2 + \dots + a_n = 2k - 1 \\ a_i \in \mathbb{Z}, a_i \geq 0, a_i \neq 1 \end{array} \right\} \quad (2)$$

Then the number of solutions for (1) and (2), denoted by $A_n(2k)$ and $A_n(2k - 1)$ respectively, satisfy $A_n(2k) = O(n^k)$, and $A_n(2k - 1) = O(n^{k-1})$.

Proof. The maximal order of the $A_n(2k)$ comes from the $\{(2, \dots, 2), (0, \dots, 0)\}$, where $(2, \dots, 2)$ is a k -tuple. There are $\binom{n}{k}$ solutions for (1) of this form. The order is $O(n^k)$, because

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!} = O(n^k).$$

The maximal order of the $A_n(2k - 1)$ comes from the $\{(2, \dots, 2, 3), (0, \dots, 0)\}$, where the k -tuple $(2, 2, \dots, 2, 3)$ has $k - 1$ elements equal to 2. There are $\binom{n}{k-1}$ solutions of (2) of this form. The order is $O(n^{k-1})$ because

$$\binom{n}{k-1} = \frac{n(n-1)\cdots(n-k+2)}{(k-1)!} = O(n^{k-1}).$$

Lemma A.5 Let X_1, X_2, \dots, X_n be independent identically distributed random variables with $E(X_i) = \mu$, and k is a positive integer. Then $E(\bar{X} - \mu)^{2k-1}$ and $E(\bar{X} - \mu)^{2k}$, if they exist, are both $O(1/n^k)$.

Proof. Without loss of generality, we suppose $E(X) = \mu = 0$, then

$$E(\bar{X}_n^{2k}) = \frac{1}{n^{2k}} E\left(\sum_{i=1}^n X_i\right)^{2k} = \frac{O(n^k)}{n^{2k}} = O(n^{-k}),$$

$$E(\bar{X}_n)^{2k-1} = \frac{1}{n^{2k-1}} E\left(\sum_{i=1}^n X_i\right)^{2k-1} = \frac{O(n^{k-1})}{n^{2k-1}} = O(n^{-k}).$$

Appendix B.

Here we present the set up we use for the linear regression case and lemmas that guarantee the validity of the obtained results.

The Gauss-Markov set up for a linear model defines $y_i = x_i^T \beta + \varepsilon_i$, where y_1, y_2, \dots, y_n are observable response variables and $X = (x_{ij})$ is an $n_1 \times p$ matrix of known constants. Moreover $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ are unobservable random variables that follow a probability distribution F , and are

such that $E(\varepsilon_i) = 0$ and $Var(\varepsilon_i) = \sigma^2$, $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ are independent. The least square solution is $\hat{\beta} = (X^T X)^{-1} X^T Y$, where Y is an $n_1 \times 1$ vector, so that $E\hat{\beta} = \beta$ and $Var(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$.

Consider an arbitrary linear combination $U_n = \lambda^T (\hat{\beta} - \beta)$, $\lambda \in \mathbb{R}^p$. Then $U = \lambda^T (X^T X)^{-1} X^T \varepsilon_i$ with $c = \lambda^T (X^T X)^{-1} X^T$. To obtain the asymptotic distribution of U all is needed is to verify that c satisfies the regularity condition of Hajek-Sidak central limit theorem.

We need first the following definition.

Definition(Convergence in distribution). A sequence $\{T_n\}$ of random variables with distributions $\{F_n\}$ is said to converge in distribution (or in law) to a (possible degenerate) random variable T with a distribution function F , if for every $\varepsilon > 0$, there exists $n_0 = n_0(\varepsilon)$, $n_0 \in \mathbf{Z}^+$ such that at every point of continuity x of F

$$|F_n(x) - F(x)| < \varepsilon,$$

for all $n \geq n_0$.

Hajek-Sidak Central Limit Theorem(Sen and Singer, 1993). Let $\{Y_n\}$ be a sequence of independent, identically distributed random variables with mean μ and variance σ^2 finite; let $\{C_n\}$ be a sequence of real vectors. Then if $C_n = (c_{n1}, c_{n2}, \dots, c_{nm})^T$ and

$$\frac{\max_{1 \leq i \leq n} c_{ni}^2}{\sum_{i=1}^n c_{ni}^2} \rightarrow 0, \text{ as } n \rightarrow +\infty$$

it follows that

$$Z_n = \frac{\sum_{i=1}^n c_{ni}(Y_i - \mu)}{\sqrt{\sigma^2 \sum_{i=1}^n c_{ni}^2}} \xrightarrow{D} Z$$

where Z is a $N(0, 1)$ random variable.

The following theorem completes the proof of the asymptotic distribution of the least squares estimator.

Cramer-Wold Theorem (Sen and Singer, 1993). Let X_1, X_2, \dots be random vectors in \mathbf{R}^p ; then $X_n \xrightarrow{D} X$ if and only if, for every fixed $\lambda \in \mathbf{R}^p$ we have $\lambda^T X_n \xrightarrow{D} \lambda^T X$.

Remark: We note here that the generalized Noether condition (assumption 2) can be modified to extend the asymptotic normality result to the heteroscedastic model, that is, the model where $E(\varepsilon) = \sigma_i^2$, $i = 1, 2, \dots, n_1$. Also notice that the normality of the least squares estimators is not obtained under normality of the errors. Assumptions 1 and 2 of section 3.2 together with the finiteness of the second moment of the, otherwise unknown, error distribution suffices for these results to hold.

The following lemmas that are listed without proof are used to arrive at the given form of the covariance terms.

Lemma B.1 Let U be distributed as a $N(0, V)$ random variable. Then

$$Var(U^T A U) = 2tr(AV)^2$$

where A is a known matrix.

Lemma B.2 Let U be distributed as a $N(\mu, V)$ random variable. Then

- (i) $E(U^T A U) = \text{tr}(A V) + \mu^T A \mu$,
- (ii) $\text{Cov}(U, U^T A U) = 2V A \mu$,
- (iii) $\text{Cov}(U^T P U, U^T Q U) = 2\text{tr}[P V Q V] + 4\mu^T P V Q \mu$.

The following lemmas are used in establishing the equivalence of the different cases in the computation of the covariance terms. The first lemma, the well-know Holder's inequality, is stated without proof.

Lemma B.3 Denote by $\|X\|_p = E^{1/p}(|X|^p)$, $p > 0$, where X is a random variable, the p -norm of X . Then, if X, Y are measurable functions on a probability space for $p > 1$, $p' > 1$, $\frac{1}{p} + \frac{1}{p'} = 1$

$$E|XY| \leq \|X\|_p \cdot \|Y\|_{p'}.$$

The special case where $p = p' = 2$ is known as Schwarz's inequality.

Lemma B.4 Let $S_j, S_{j'}$ be training sets and $S_j^c, S_{j'}^c$ their corresponding test sets. Assume that for $(y_i, x_i) \in S_j^c$, $(y_i, x_i) \in S_j$, for some $i \in \{1, 2, \dots, n_2\}$. Assume that $E([L'(x_i^T \beta_0, y_i)]^2) < \infty$, and $E[L^4(x_i^T \beta_0, y_i)] < \infty$,

$$\begin{aligned} \sup_{\|\hat{\beta}_{S_{j'}} - \beta_0\| \leq k/\sqrt{n_1}} & |E[L(x_i \beta_0, y_i) L'(x_i^T \beta_0, y_i) x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] \\ & - E[L(x_i \beta_0, y_i)] E[L'(x_i^T \beta_0, y_i)] E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| = o(1) \end{aligned}$$

Proof Write

$$\begin{aligned} & |E[L(x_i \beta_0, y_i) L'(x_i^T \beta_0, y_i) x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] - E[L(x_i \beta_0, y_i)] E[L'(x_i^T \beta_0, y_i)] E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\ & \leq |E[L(x_i \beta_0, y_i) L'(x_i^T \beta_0, y_i) x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| + |E[L(x_i \beta_0, y_i)] E[L'(x_i^T \beta_0, y_i)] E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\ & \leq E\{|L(x_i \beta_0, y_i) x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)| |L'(x_i^T \beta_0, y_i)|\} + |E[L(x_i \beta_0, y_i)]| |E[L'(x_i^T \beta_0, y_i)]| |E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \end{aligned}$$

Using lemma A2.3 and the fact that $E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] = 0$ the above relationship becomes:

$$\begin{aligned} & |E[L(x_i \beta_0, y_i) L'(x_i^T \beta_0, y_i) x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] - E[L(x_i \beta_0, y_i)] E[L'(x_i^T \beta_0, y_i)] E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\ & \leq \sqrt{E([L(x_i^T \beta_0, y_i)]^2)} \sqrt{E[L^2(x_i^T \beta_0, y_i) (\hat{\beta}_{S_{j'}} - \beta_0) x_i x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]} \end{aligned}$$

Apply once more Lemma A2.3 on

$$\begin{aligned} & E[L^2(x_i^T \beta_0, y_i) (\hat{\beta}_{S_{j'}} - \beta_0) x_i x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] \\ & \leq \sqrt{E[L^4(x_i^T \beta_0, y_i)]} \sqrt{E[(\hat{\beta}_{S_{j'}} - \beta_0) x_i x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]} \end{aligned}$$

Thus

$$\begin{aligned}
 & \sup_{\|\hat{\beta}_{S_{j'}} - \beta_0\| \leq k/\sqrt{n_1}} |E[L(x_i \beta_0, y_i) L'(x_i^T \beta_0, y_i) x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] \\
 & - E[L(x_i \beta_0, y_i)] E[L'(x_i^T \beta_0, y_i)] E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\
 & \leq \sup_{\|\hat{\beta}_{S_{j'}} - \beta_0\| \leq k/\sqrt{n_1}} M \cdot \sqrt[4]{E[(\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]^2} \\
 & \leq M \cdot \sqrt[4]{\left[\left(\sum_{l=1}^p x_{i',l} \right)^2 \frac{k^2}{p^2 n_1} \right]^2} \\
 & \leq \frac{1}{\sqrt{n_1}} \left[\frac{M_k}{p} \left(\sum_{l=1}^p x_{i',l} \right) \right] \\
 & \leq \frac{c}{\sqrt{n_1}}
 \end{aligned}$$

where $c = \frac{M_k}{p} \left(\sum_{l=1}^p x_{i',l} \right) < \infty$.

Lemma B.5 Let $S_j, S_{j'}$ be two training sets and $S_j^c, S_{j'}^c$ be their corresponding test sets. Under the assumption that $E[L''(x_i^T \beta_0, y_i)]$ is finite and for some $(y_i, x_i) \in S_j^c, (y_i, x_i) \in S_{j'}^c$.

$$\begin{aligned}
 & \sup_{\|\hat{\beta}_{S_{j'}} - \beta_0\| \leq k/\sqrt{n_1}} |E[L(x_i^T \beta_0, y_i) L''(x_i^T \beta_0, y_i) (\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] \\
 & - E[L(x_i^T \beta_0, y_i)] E[L''(x_i^T \beta_0, y_i)] E[(\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| = o(1)
 \end{aligned}$$

Proof. Write

$$\begin{aligned}
 & |E[L(x_i^T \beta_0, y_i) L''(x_i^T \beta_0, y_i) (\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)] \\
 & - E[L(x_i^T \beta_0, y_i)] E[L''(x_i^T \beta_0, y_i)] E[(\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\
 & \leq |E[L(x_i^T \beta_0, y_i) L''(x_i^T \beta_0, y_i) (\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\
 & + |E[L(x_i^T \beta_0, y_i)] E[L''(x_i^T \beta_0, y_i)] E[(\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]|
 \end{aligned}$$

The first term of the above relationship gives:

$$\begin{aligned}
 & |E[L(x_i^T \beta_0, y_i) L''(x_i^T \beta_0, y_i) (\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\
 & \leq E\{ |L(x_i^T \beta_0, y_i) (\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)| L''(x_i^T \beta_0, y_i) \} \\
 & \leq \sqrt{E[L''(x_i^T \beta_0, y_i)^2]} \sqrt{E[L^2(x_i^T \beta_0, y_i) ((\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)^2)]} \\
 & \leq \frac{c}{n_1}
 \end{aligned}$$

where c is a constant. The second term is

$$\begin{aligned}
 & |E[L(x_i^T \beta_0, y_i)] E[L''(x_i^T \beta_0, y_i)] E[(\hat{\beta}_{S_{j'}} - \beta_0) x_i^T x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]| \\
 & \leq E[|L(x_i^T \beta_0, y_i) L''(x_i^T \beta_0, y_i)|] E[x_i^T (\hat{\beta}_{S_{j'}} - \beta_0)]^2 \\
 & \leq |E[L(x_i^T \beta_0, y_i)]| E[|L''(x_i^T \beta_0, y_i)|] \frac{c_1}{n_1} \\
 & \leq \frac{c^*}{n_1}
 \end{aligned}$$

where c^* is a constant. Thus the lemma is proved. Similarly we can prove that the terms, in the computation of covariance, where $(y_i, x_i) \in S_j$ and/or $(y_{i'}, x_{i'}) \in S_j$ can be replaced and treated as the case where $(y_i, x_i) \notin S_j$ and/or $(y_{i'}, x_{i'}) \notin S_j$ in the neighborhood of the true value of β_0 .

Lemma B.6 Suppose

$$x = \begin{pmatrix} u \\ v \end{pmatrix} \sim MVN \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \right)$$

where u is $q \times 1$ vector, v is $(p-q) \times 1$ vector, a is a known $q \times 1$ vector, B is known $(p-q) \times (p-q)$ matrix.

Then

$$E(a^T uv^T Bv) = 0.$$

Proof: Using conditional probability argument, we have

$$\begin{aligned} E(a^T uv^T Bv) &= E_u \{ E_{v|u} [a^T uv^T Bv] \} \\ &= E_u \{ a^T u E_{v|u} [v^T Bv] \} \\ &= E_u \{ a^T u [tr(B\Sigma_{22.1}) - (\Sigma_{21}\Sigma_{11}^{-1}u)^T B(\Sigma_{21}\Sigma_{11}^{-1}u)] \} \\ &= E_u \{ a^T u (\Sigma_{21}\Sigma_{11}^{-1}u)^T B(\Sigma_{21}\Sigma_{11}^{-1}u) \} \\ &= E_u \{ a^T uu^T \Sigma_{11}^{-1} \Sigma_{12} B(\Sigma_{21}\Sigma_{11}^{-1}u) \} \\ &= E_u \{ a^T uu^T Cu \} \\ &= a^T E_u \{ uu^T Cu \} \\ &= a^T \{ Cov(u, u^T Cu) + EuE(u^T Cu) \} \\ &= a^T 2\Sigma_{22}C \cdot 0 + 0 \\ &= 0 \end{aligned}$$

where $c = \Sigma_{11}^{-1} \Sigma_{12} B \Sigma_{21} \Sigma_{11}^{-1}$. We use the property that if x is $N(\mu, V)$, then $cov(x, x^T Ax) = 2vA\mu$.

Appendix C.

Proof of Proposition 3.5: To obtain the approximation given above we need first an approximation for the product $L(\hat{y}_{i,S_j}, y_i)L(\hat{y}_{i',S_{j'}}, y_{i'})$. Using expansion (3.13) we obtain:

$$\begin{aligned}
 L(\hat{y}_{i,S_j}, y_i)L(\hat{y}_{i',S_{j'}}, y_{i'}) &= L(x_i\beta_0, y_i)L(x_{i'}^T\beta_0, y_{i'}) + L(x_i\beta_0, y_i)L'(x_{i'}^T\beta_0, y_{i'})x_{i'}^T(\hat{\beta}_{S_j} - \beta_0) \\
 &+ \frac{1}{2}L(x_i\beta_0, y_i)L''(x_{i'}^T\beta_0, y_{i'})x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0)^T x_{i'}x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0) \\
 &+ L'(x_i\beta_0, y_i)x_i^T(\hat{\beta}_{S_j} - \beta_0)L(x_{i'}^T\beta_0, y_{i'}) \\
 &+ L'(x_i\beta_0, y_i)x_i^T(\hat{\beta}_{S_j} - \beta_0)L'(x_{i'}^T\beta_0, y_{i'})x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0) \\
 &+ \frac{1}{2}L(x_i^T\beta_0, y_i)x_i^T(\hat{\beta}_{S_j} - \beta_0)L'(x_{i'}^T\beta_0, y_{i'})(\hat{\beta}_{S_{j'}} - \beta_0)x_{i'}x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0) \\
 &+ \frac{1}{2}L(x_{i'}^T\beta_0, y_{i'})L''(x_i^T\beta_0, y_i)(\hat{\beta}_{S_j} - \beta_0)x_ix_i^T(\hat{\beta}_{S_j} - \beta_0) \\
 &+ \frac{1}{2}L'(x_{i'}^T\beta_0, y_{i'})x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0)L''(x_i^T\beta_0, y_i)(\hat{\beta}_{S_j} - \beta_0)x_ix_i^T(\hat{\beta}_{S_j} - \beta_0) \\
 &+ \frac{1}{4}L''(x_i^T\beta_0, y_i)L''(x_{i'}^T\beta_0, y_{i'})(\hat{\beta}_{S_j} - \beta_0)x_ix_i^T(\hat{\beta}_{S_j} - \beta_0) \\
 &\quad (\hat{\beta}_{S_{j'}} - \beta_0)^T x_{i'}x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0) + R_n. \tag{1}
 \end{aligned}$$

We need the expectation, over everything random, of relationship (1). Assume first that $i \neq i'$. Recall that $(y_i, x_i) \in S_j^c$ and $(y_{i'}, x_{i'}) \in S_{j'}^c$ and (y_i, x_i) is independent of $(y_{i'}, x_{i'})$. Then the first term of the above expansion is

$$E[L(x_i\beta_0, y_i)]E[L(x_{i'}^T\beta_0, y_{i'})] = (E[L(x_i\beta_0, y_i)])^2. \tag{2}$$

(If $L(x_i^T\beta, y_i) = (x_i^T\beta_0 - y_i)^2 = \varepsilon_i^2$ and the $E(\varepsilon_i^2) = \sigma^2$).

We need now

$$E\{L(x_i^T\beta_0, y_i)L'(x_{i'}^T\beta_0, y_{i'})x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0)\} \tag{3}$$

Notice that all expectations here are conditional on X , that is, we treat the fixed design case. To evaluate this expectation we need to distinguish between two cases. The first corresponding to $(y_i, x_i) \notin S_{j'}$. In this case (3) equals 0. The second corresponds to $(y_i, x_i) \in S_{j'}$. Lemma B.4 of the appendix proves that (3) can be replaced by

$$E[L(x_i^T\beta_0, y_i)]E[L'(x_{i'}^T\beta_0, y_{i'})]E[x_{i'}^T(\hat{\beta}_{S_{j'}} - \beta_0)] = 0. \tag{4}$$

Therefore the second term is 0. Similarly, the expectation of the third term is

$$\frac{\sigma^2}{2}E[L(x_i^T\beta_0, y_i)]E[L''(x_{i'}^T\beta_0, y_{i'})]tr[(x_{i'}x_{i'}^T)(\mathbf{X}_{S_{j'}}^T\mathbf{X}_{S_{j'}})^{-1}], \tag{5}$$

in both cases, when $(y_i, x_i) \notin S_{j'}$ and when $(y_i, x_i) \in S_{j'}$.

The expectation of the fourth term of relationship (1) is 0. To evaluate the expectation of the fifth term we distinguish four cases: (i) $(y_i, x_i) \notin S_{j'}$ and $(y_{i'}, x_{i'}) \notin S_j$, (ii) $(y_i, x_i) \notin S_{j'}$ and $(y_{i'}, x_{i'}) \in S_j$,

(iii) $(y_i, x_i) \in S_{j'}$ and $(y_{i'}, x_{i'}) \notin S_j$, (iv) $(y_i, x_i) \in S_{j'}$ but $(y_{i'}, x_{i'}) \in S_j$. Lemma B.6 of the appendix allows in case (ii), (iii) and (iv), the replacement of the correct value of the expectation by the value obtained from expression (6) given below. Thus, the expectation of the fifth term is:

$$E[L'(x_i^T \beta_0, y_i)]E[L'(x_{i'}^T \beta_0, y_{i'})]x_i^T Cov(\hat{\beta}_{S_j}, \hat{\beta}_{S_{j'}})x_{i'}. \quad (6)$$

Since $S_j \cap S_{j'} \neq \emptyset$, and assuming the $X_{S_j}, X_{S_{j'}}$ have that upper $k \times p$ part common, relationship (.6) can be written as

$$\sigma^2(E[L'(x_i^T \beta_0, y_i)])^2 x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} x_{i'},$$

where X_1 is of dimension $k \times p$, $k = Card(S_j \cap S_{j'})$, and σ^2 is the population variance. To compute the expectation of the sixth term we again distinguish between case (i), (ii), (iii) and (iv) as above. However, all cases reduce to the case (i). For this expectation we have from lemma B.6,

$$\frac{1}{2}(E[L'(x_i^T \beta_0, y_i)])E[L''(x_{i'}^T \beta_0, y_{i'})]E[x_i^T (\hat{\beta}_{S_j} - \beta_0)(\hat{\beta}_{S_{j'}} - \beta_0)x_{i'}^T (\hat{\beta}_{S_{j'}} - \beta_0)] = 0. \quad (7)$$

For the expectation of the seventh term we distinguish two cases: (i) $(y_{i'}, x_{i'}) \notin S_j$ and (ii) $(y_{i'}, x_{i'}) \in S_j$. Both cases can be treated using the following expression for the expectation of the seventh term:

$$\begin{aligned} & \frac{\sigma^2}{2}E[L(x_{i'}^T \beta_0, y_{i'})](E[L''(x_i^T \beta_0, y_i)])E[(\hat{\beta}_{S_j} - \beta_0)x_i x_i^T (\hat{\beta}_{S_j} - \beta_0)] \\ &= \frac{\sigma^2}{2}E[L(x_{i'}^T \beta_0, y_{i'})](E[L''(x_i^T \beta_0, y_i)])tr[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}]. \end{aligned} \quad (8)$$

The expectation of the eighth term is treated as the expectation of the sixth term, therefore it is given by relationship (7). For the expectation of last term we distinguish the four different cases that are listed above. In this case again all different cases can be treated as case (i). Therefore the expectation of the ninth term is

$$\frac{1}{4}E[L''(x_i^T \beta_0, y_i)]^2 E[(\hat{\beta}_{S_j} - \beta_0)^T x_i x_i^T (\hat{\beta}_{S_j} - \beta_0)(\hat{\beta}_{S_{j'}} - \beta_0)^T x_{i'} x_{i'}^T (\hat{\beta}_{S_{j'}} - \beta_0)] \quad (9)$$

But

$$\begin{aligned} & E[(\hat{\beta}_{S_j} - \beta_0)^T x_i x_i^T (\hat{\beta}_{S_j} - \beta_0)(\hat{\beta}_{S_{j'}} - \beta_0)^T x_{i'} x_{i'}^T (\hat{\beta}_{S_{j'}} - \beta_0)] \\ &= 2tr[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}(x_{i'} x_{i'}^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}] \\ & \quad + \sigma^4 tr[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \cdot tr[(x_{i'} x_{i'}^T)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}]. \end{aligned} \quad (10)$$

Therefore the covariance is given as

$$\begin{aligned} & Cov(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_{j'}}, y_{i'})) = \sigma^2(E[L'(x_i^T \beta_0, y_i)])^2 x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1} x_{i'} \\ & + \frac{\sigma^4}{2}(E[L''(x_i^T \beta_0, y_i)])^2 tr((x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}(x_{i'} x_{i'}^T) \\ & \quad (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_{j'}}^T \mathbf{X}_{S_{j'}})^{-1}). \end{aligned}$$

Note that, when L is the square error loss the covariance is given as

$$2\sigma^4 \text{tr}\{(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1}(x_{i'} x_{i'}^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1}\}.$$

When $i = i'$, the covariance is given as

$$\begin{aligned} \text{Cov}(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_j}, y_{i'})) &= \text{Var}(L(x_i^T \beta_0, y_i)) + \frac{\sigma^2}{2} \text{Cov}(L(x_i^T \beta_0, y_i), L''(x_i^T \beta_0, y_i)) \\ &\quad (x_i^T (\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1} x_i + x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i) \\ &+ \sigma^2 (E[L'(x_i^T \beta_0, y_i)])^2 x_i^T (\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i \\ &+ \frac{\sigma^4}{2} (E[L''(x_i^T \beta_0, y_i)])^2 \text{tr}((x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1}(x_i x_i^T) \\ &\quad (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1}) \\ &+ \frac{\sigma^4}{4} \text{Var}(L''(x_i^T \beta_0, y_i)) x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i x_i^T (\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1} x_i. \end{aligned}$$

Note that, when L is the square error loss the covariance is given as

$$\begin{aligned} &2\sigma^4 + 4\sigma^4 x_i^T (\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1} (\mathbf{X}_1^T \mathbf{X}_1) (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i \\ &+ 2\sigma^4 \text{tr}\{(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1}(x_{i'} x_{i'}^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}(\mathbf{X}_1^T \mathbf{X}_1)(\mathbf{X}_{S_j'}^T \mathbf{X}_{S_j'})^{-1}\} \end{aligned}$$

Proof of Proposition 3.6: Write:

$$\begin{aligned} L(\hat{y}_{i,S_j}, y_i) L(\hat{y}_{i',S_j}, y_{i'}) &= L(x_i \beta_0, y_i) L(x_{i'}^T \beta_0, y_{i'}) + L(x_i \beta_0, y_i) L'(x_{i'}^T \beta_0, y_{i'}) x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ \frac{1}{2} L(x_i \beta_0, y_i) L''(x_{i'}^T \beta_0, y_{i'}) (\hat{\beta}_{S_j} - \beta_0)^T x_{i'} x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ L'(x_i \beta_0, y_i) L(x_{i'}^T \beta_0, y_{i'}) x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ L'(x_i \beta_0, y_i) x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) L'(x_{i'}^T \beta_0, y_{i'}) x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ \frac{1}{2} L'(x_i^T \beta_0, y_i) L''(x_{i'}^T \beta_0, y_{i'}) x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) (\hat{\beta}_{S_j} - \beta_0) x_{i'} x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ \frac{1}{2} L(x_{i'}^T \beta_0, y_{i'}) L''(x_i^T \beta_0, y_i) (\hat{\beta}_{S_j} - \beta_0) x_i x_i^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ \frac{1}{2} L'(x_{i'}^T \beta_0, y_{i'}) L''(x_i^T \beta_0, y_i) x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) (\hat{\beta}_{S_j} - \beta_0) x_i x_i^T (\hat{\beta}_{S_j} - \beta_0) \\ &+ \frac{1}{4} L''(x_i^T \beta_0, y_i) L''(x_{i'}^T \beta_0, y_{i'}) (\hat{\beta}_{S_j} - \beta_0) x_i x_i^T (\hat{\beta}_{S_j} - \beta_0) \\ &\quad (\hat{\beta}_{S_j} - \beta_0)^T x_{i'} x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) + R_n. \end{aligned} \tag{11}$$

We need to evaluate the expectation of relation (11). We have

$$\begin{aligned}
 & E\{L(\hat{y}_{i,S_j}, y_i)L(\hat{y}_{i',S_j}, y_{i'})\} \\
 = & (E[L(x_i^T \beta_0, y_i)])^2 + \frac{\sigma^2}{2} E[L(x_i^T \beta_0, y_i)]E[L''(x_i^T \beta_0, y_i)] \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \sigma^2 E[L'(x_i^T \beta_0, y_i)]^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \frac{\sigma^2}{2} E[L(x_i^T \beta_0, y_i)]E[L''(x_i^T \beta_0, y_i)] \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \frac{1}{2} E[L'(x_i^T \beta_0, y_i)]E[L''(x_i^T \beta_0, y_i)]E[x_i^T (\hat{\beta}_{S_j} - \beta_0)(\hat{\beta}_{S_j} - \beta_0) x_i x_i^T (\hat{\beta}_{S_j} - \beta_0)] \\
 + & \frac{1}{2} E[L'(x_{i'}^T \beta_0, y_{i'})]E[L''(x_{i'}^T \beta_0, y_{i'})]E[x_{i'}^T (\hat{\beta}_{S_j} - \beta_0)(\hat{\beta}_{S_j} - \beta_0) x_{i'} x_{i'}^T (\hat{\beta}_{S_j} - \beta_0)] \\
 + & \frac{1}{4} (E[L''(x_i^T \beta_0, y_i)])^2 E[(\hat{\beta}_{S_j} - \beta_0) x_i x_i^T (\hat{\beta}_{S_j} - \beta_0)(\hat{\beta}_{S_j} - \beta_0) x_{i'} x_{i'}^T (\hat{\beta}_{S_j} - \beta_0)].
 \end{aligned}$$

Now,

$$\begin{pmatrix} x_i^T (\hat{\beta}_{S_j} - \beta_0) \\ x_{i'}^T (\hat{\beta}_{S_j} - \beta_0) \end{pmatrix} \sim MVN\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma\right) \quad (12)$$

where

$$\Sigma = \sigma^2 \begin{pmatrix} x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i & x_i^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_{i'} \\ x_{i'}^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_i & x_{i'}^T (\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} x_{i'} \end{pmatrix}.$$

Notice here that we do not assume normality of the errors. The assumption of normality for the error distribution is too restrictive. Instead, assumptions A1 and A2 establish the asymptotic distribution of the least squares estimators as the size of the training set n_1 becomes larger and larger. That guarantees that (12) holds. Therefore,

$$\begin{aligned}
 & E\{L(\hat{y}_{i,S_j}, y_i)L(\hat{y}_{i',S_j}, y_{i'})\} \\
 = & (E[L(x_i^T \beta_0, y_i)])^2 + \frac{\sigma^2}{2} E[L(x_i^T \beta_0, y_i)]E[L''(x_i^T \beta_0, y_i)] \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \sigma^2 E[L'(x_i^T \beta_0, y_i)]^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \frac{\sigma^2}{2} E[L(x_i^T \beta_0, y_i)]E[L''(x_i^T \beta_0, y_i)] \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \frac{\sigma^4}{2} (E[L''(x_i^T \beta_0, y_i)])^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \frac{\sigma^4}{4} (E[L''(x_i^T \beta_0, y_i)])^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \text{tr}[(x_{i'} x_{i'}^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 = & (E[L(x_i^T \beta_0, y_i)])^2 + \frac{\sigma^2}{2} E[L(x_i^T \beta_0, y_i)]E[L''(x_i^T \beta_0, y_i)] (\text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \text{tr}[(x_{i'} x_{i'}^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}]) + \sigma^2 E[L'(x_i^T \beta_0, y_i)]^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \frac{\sigma^4}{2} (E[L''(x_i^T \beta_0, y_i)])^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\
 + & \frac{\sigma^4}{4} (E[L''(x_i^T \beta_0, y_i)])^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \text{tr}[(x_{i'} x_{i'}^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}].
 \end{aligned}$$

Therefore,

$$\begin{aligned} \text{Cov}(L(\hat{y}_{i,S_j}, y_i), L(\hat{y}_{i',S_j}, y_{i'})) &= \sigma^2 (E[L'(x_i^T \beta_0, y_i)])^2 \text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}] \\ + \frac{\sigma^4}{2} (E[L''(x_i^T \beta_0, y_i)])^2 &\text{tr}[(x_i x_i^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1} (x_{i'} x_{i'}^T)(\mathbf{X}_{S_j}^T \mathbf{X}_{S_j})^{-1}]. \end{aligned}$$

References

Y. Bengio and Y. Grandvalet. No unbiased estimator of the variance of k -fold cross validation. *Journal of Machine Learning Research*, **5**: 1089-1105, 2004.

P. J. Bickel and K. A. Doksum. *Mathematical Statistics*, Prentice Hall, 2001.

L. Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, **24**: 2350-2383, 1996.

H. Cramer. *Mathematical Methods of Statistics*. Princeton University Press, 19th Printing, 1999.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, **10**: 1895-1923, 1998.

B. Efron. Estimating the error rate of a predication rule: Improvement on cross-validation. *Journal of the American Statistical Association*, **78**: 316-331, 1983.

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

B. Efron. The Estimation of Prediction Error: Covariance Penalties and Cross-Validation. *Journal of the American Statistical Association*, **99**: 619-632, 2004.

T. Hastie, R. Tibshirani and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, 2001.

K. Hitomi and M. Kagihara. Calculation methods for nonlinear dynamic least absolute deviations estimator. *Journal of the Japan. Statist. Society*, **31**: 39-51, 2001.

A. D. Ioffe and J-P. Penot. Limiting subhessians, limiting subjets and their calculus. *Transactions of the American Mathematical Society*, **349**: 789-807, 1997.

G. M. James. Variance and bias for general loss functions. *Machine Learning*, **51**: 115-135, 2003.

M. Kearns. A bound on the error of cross validation with consequences for the training-test split. *In Advances in Neural Information Processing Systems*, **8**: 183-189, 1996.

- M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross validation. *Neural Computation*, **11**: 1427-1453, 1999.
- R. A. Khan. Approximation of the expectation of a function of the sample mean. *Statistics*, **38**: 117-122, 2004.
- R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *In The International Joint Conference on Artificial Intelligence*, 1137-1143, 1995.
- E. L. Lehmann. *Theory of Point Estimation*. Wiley and Sons, 1983.
- G. J. McLachlan. An asymptotic expansion for the variance of the errors of misclassification of the linear discriminant function. *Australian Journal of Statistics*, **14**: 68-72, 1972.
- G. J. McLachlan. An asymptotic expansion of the expectation of the estimated error rate in discriminant analysis. *Australian Journal of Statistics*, **15**: 210-214, 1974.
- G. J. McLachlan. The asymptotic distributions of the conditional error rate and risk in discriminant analysis. *Biometrika*, **61**: 131-135, 1974.
- G. J. McLachlan. The bias of the apparent error rate in discriminant analysis. *Biometrika*, **63**: 239-244, 1976.
- C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, **52**: 239-281, 2003.
- R. R. Picard and R. D. Cook. Cross validation of regression models. *Journal of the American Statistical Association*, **79**: 575-583, 1984.
- J. Piper. Variability and bias in experimentally measured classifier error rates. *Pattern Recognition Letters*, **13**: 685-692, 1992.
- E. Ronchetti and L. Ventura. Between stability and higher order asymptotics. *Statistics and Computing*, **11**: 67-73, 2001.
- P. K. Sen and J. M. Singer. *Large Sample Methods in Statistics: An Introduction with Applications*. Chapman and Hall, 1993.

Semigroup Kernels on Measures

Marco Cuturi

*Ecole des Mines de Paris
35 rue Saint Honoré
77305 Fontainebleau, France;
Institute of Statistical Mathematics
4-6-7 Minami-Azabu, Minato-Ku, Tokyo, Japan*

MARCO.CUTURI@ENSMP.FR

Kenji Fukumizu

*Institute of Statistical Mathematics
4-6-7 Minami-Azabu, Minato-Ku, Tokyo, Japan*

FUKUMIZU@ISM.AC.JP

Jean-Philippe Vert

*Ecole des Mines de Paris
35 rue Saint Honoré
77305 Fontainebleau, France*

JEAN-PHILIPPE.VERT@ENSMP.FR

Editor: John Lafferty

Abstract

We present a family of positive definite kernels on measures, characterized by the fact that the value of the kernel between two measures is a function of their sum. These kernels can be used to derive kernels on structured objects, such as images and texts, by representing these objects as sets of components, such as pixels or words, or more generally as measures on the space of components. Several kernels studied in this work make use of common quantities defined on measures such as entropy or generalized variance to detect similarities. Given an a priori kernel on the space of components itself, the approach is further extended by restating the previous results in a more efficient and flexible framework using the “kernel trick”. Finally, a constructive approach to such positive definite kernels through an integral representation theorem is proved, before presenting experimental results on a benchmark experiment of handwritten digits classification to illustrate the validity of the approach.

Keywords: kernels on measures, semigroup theory, Jensen divergence, generalized variance, reproducing kernel Hilbert space

1. Introduction

The challenge of performing classification or regression tasks over complex and non vectorial objects is an increasingly important problem in machine learning, motivated by diverse applications such as bioinformatics or multimedia document processing. The kernel method approach to such problems (Schölkopf and Smola, 2002) is grounded on the choice of a proper similarity measure, namely a positive definite (p.d.) kernel defined between pairs of objects of interest, to be used alongside with kernel methods such as support vector machines (Boser et al., 1992). While natural similarities defined through dot-products and related distances are available when the objects lie in a Hilbert space, there is no standard dot-product to compare strings, texts, videos, graphs or other

structured objects. This situation motivates the proposal of various kernels, either tuned and trained to be efficient on specific applications or useful in more general cases.

One possible approach to kernel design for such complex objects consists in representing them by sets of basic components easier to manipulate, and designing kernels on such sets. Such basic components can typically be subparts of the original complex objects, obtained by exhaustive enumeration or random sampling. For example, a very common way to represent a text for applications such as text classification and information retrieval is to break it into words and consider it as a bag of words, that is, a finite set of weighted terms. Another possibility is to extract all fixed-length blocks of consecutive letters and represent the text by the vector of counts of all blocks (Leslie et al., 2002), or even to add to this representation additional blocks obtained by slight modifications of the blocks present in the text with different weighting schemes (Leslie et al., 2003). Similarly, a grey-level digitalized image can be considered as a finite set of points of \mathbb{R}^3 where each point (x, y, I) stands for the intensity I displayed on the pixel (x, y) in that image (Kondor and Jebara, 2003).

Once such a representation is obtained, different strategies have been adopted to design kernels on these descriptions of complex objects. When the set of basic components is finite, this representation amounts to encode a complex object as a finite-dimensional vector of counters, and any kernel for vectors can be then translated to a kernel for complex object through this feature representation (Joachims, 2002, Leslie et al., 2002, 2003). For more general situations, several authors have proposed to handle such weighted lists of points by first fitting a probability distribution to each list, and defining a kernel between the resulting distributions (Lafferty and Lebanon, 2002, Jebara et al., 2004, Kondor and Jebara, 2003, Hein and Bousquet, 2005). Alternatively, Cuturi and Vert (2005) use a parametric family of densities and a Bayesian framework to define a kernel for strings based on the mutual information between their sets of variable-length blocks, using the concept of mutual information kernels (Seeger, 2002). Finally, Wolf and Shashua (2003) recently proposed a formulation rooted in kernel canonical correlation analysis (Bach and Jordan, 2002, Melzer et al., 2001, Akaho, 2001) which makes use of the principal angles between the subspaces generated by the two sets of points to be compared when considered in a feature space.

We explore in this contribution a different direction to kernel design for weighted lists of basic components. Observing that such a list can be conveniently represented by a molecular measure on the set of basic components, that is a weighted sum of Dirac measures, or that the distribution of points might be fitted by a statistical model and result in a density on the same set, we formally focus our attention on the problem of defining a kernel between finite measures on the space of basic components. More precisely, we explore the set of kernels between measures that can be expressed as a function of their sum, that is:

$$k(\mu, \mu') = \varphi(\mu + \mu'). \quad (1)$$

The rationale behind this formulation is that if two measures or sets of points μ and μ' overlap, then it is expected that the sum $\mu + \mu'$ is more concentrated and less scattered than if they do not. As a result, we typically expect φ to quantify the dispersion of its argument, increasing when it is more concentrated. This setting is therefore a broad generalization of the observation by Cuturi and Vert (2005) that a valid kernel for strings, seen as bags of variable-length blocks, is obtained from the compression rate of the *concatenation* of the two strings by a particular compression algorithm.

The set of measures endowed with the addition is an Abelian semigroup, and the kernel (1) is exactly what Berg et al. (1984) call a *semigroup kernel*. The main contribution of this paper is to present several valid positive definite (p.d.) semigroup kernels for molecular measures or

densities. As expected, we prove that several functions ϕ that quantify the dispersion of measures through their entropy or through their variance matrix result in valid p.d. kernels. Using entropy to compare two measures is not a new idea (Rao, 1987) but it was recently restated within different frameworks (Hein and Bousquet, 2005, Endres and Schindelin, 2003, Fuglede and Topsøe, 2004). We introduce entropy in this paper slightly differently, noting that it is a semigroup negative definite function defined on measures. On the other hand, the use of generalized variance to derive a positive definite kernel between measures as proposed here is new to our knowledge. We further show how such kernels can be applied to molecular measures through regularization operations. In the case of the kernel based on the spectrum of the variance matrix, we show how it can be applied implicitly for molecular measures mapped to a reproducing kernel Hilbert space when a p.d. kernel on the space of basic components is provided, thanks to an application of the “kernel trick”.

Besides these examples of practical relevance, we also consider the question of characterizing *all* functions ϕ that lead to a p.d. kernel through (1). Using the general theory of semigroup kernels we state an integral representation of such kernels and study the semicharacters involved in this representation. This new result provides a constructive characterization of such kernels, which we briefly explore by showing that Bayesian mixtures over exponential models can be seen as natural functions ϕ that lead to p.d. kernels, thus making the link with the particular case treated by Cuturi and Vert (2005).

This paper is organized as follows. We first introduce elements of measure representations of weighted lists and define the semigroup formalism and the notion of semigroup p.d. kernel in Section 2. Section 3 contains two examples of semigroup p.d. kernels, which are however usually not defined for molecular measures: the entropy kernel and the inverse generalized variance (IGV) kernel. Through regularization procedures, practical applications of such kernels on molecular measures are proposed in Section 4, and the approach is further extended by kernelizing the IGV through an a priori kernel defined itself on the space of components in Section 5. Section 6 contains the general integral representation of semigroup kernels and Section 7 makes the link between p.d. kernels and Bayesian posterior mixture probabilities. Finally, Section 8 contains an empirical evaluation of the proposed kernels on a benchmark experiment of handwritten digits classification.

2. Notations and Framework: Semigroup Kernels on Measures

In this section we set up the framework and notations of this paper, in particular the idea of semigroup kernel on the semigroup of measures.

2.1 Measures on Basic Components

We model the space of basic components by a Hausdorff space (X, \mathcal{B}, ν) endowed with its Borel σ -algebra and a Borel dominant measure ν . A positive Radon measure μ is a positive Borel measure which satisfies (i) $\mu(C) < +\infty$ for any compact subset $C \subseteq X$ and (ii) $\mu(B) = \sup\{\mu(C) \mid C \subseteq B, C \text{ compact}\}$ for any $B \in \mathcal{B}$ (see for example Berg et al. (1984) for the construction of Radon measures on Hausdorff spaces). The set of positive bounded (i.e., $\mu(X) < +\infty$) Radon measures on X is denoted by $M_+^b(X)$. We introduce the subset of $M_+^b(X)$ of molecular (or atomic) measures $\text{Mol}_+(X)$, namely measures such that

$$\text{supp}(\mu) \stackrel{\text{def}}{=} \{x \in X \mid \mu(U) > 0, \text{ for all open subset } U \text{ s.t. } x \in U\}$$

is finite, and we denote by $\delta_x \in \text{Mol}_+(\mathcal{X})$ the molecular (Dirac) measure of weight 1 on x . For a molecular measure μ , an *admissible base* of μ is a finite list γ of weighted points of \mathcal{X} , namely $\gamma = (x_i, a_i)_{i=1}^d$, where $x_i \in \mathcal{X}$ and $a_i > 0$ for $1 \leq i \leq d$, such that $\mu = \sum_{i=1}^d a_i \delta_{x_i}$. We write in that case $|\gamma| = \sum_{i=1}^d a_i$ and $l(\gamma) = d$. Reciprocally, a measure μ is said to be the image measure of a list of weighted elements γ if the previous equality holds. Finally, for a Borel measurable function $f \in \mathbb{R}^{\mathcal{X}}$ and a Borel measure μ , we write $\mu[f] = \int_{\mathcal{X}} f d\mu$.

2.2 Semigroups and Sets of Points

We follow in this paper the definitions found in Berg et al. (1984), which we now recall. An *Abelian semigroup* $(S, +)$ is a nonempty set S endowed with an *associative* and *commutative composition* $+$ and a neutral element 0. Referring further to the notations used in Berg et al. (1984), note that we will only use auto-involutive semigroups in this paper, and will hence not discuss other semigroups which admit different involutions.

A function $\varphi : S \rightarrow \mathbb{R}$ is called a *positive definite* (resp. *negative definite*, n.d.) function on the semigroup $(S, +)$ if $(s, t) \mapsto \varphi(s + t)$ is a p.d. (resp. n.d.) kernel on $S \times S$. The symmetry of the kernel being ensured by the commutativity of $+$, the positive definiteness is equivalent to the fact that the inequality

$$\sum_{i,j=1}^N c_i c_j \varphi(x_i + x_j) \geq 0$$

holds for any $N \in \mathbb{N}$, $(x_1, \dots, x_N) \in S^N$ and $(c_1, \dots, c_n) \in \mathbb{R}^N$. Using the same notations, and adding the additional condition that $\sum_{i=1}^n c_i = 0$ yields the definition of negative definiteness as φ satisfying now

$$\sum_{i,j=1}^N c_i c_j \varphi(x_i + x_j) \leq 0.$$

Hence semigroup kernels are real-valued functions φ defined on the set of interest S , the similarity between two elements s, t of S being just the value taken by that function on their composition, namely $\varphi(s + t)$.

Recalling our initial goal to quantify the similarity between two complex objects through finite weighted lists of elements in \mathcal{X} , we note that $(\mathcal{P}(\mathcal{X}), \cup)$ the set of subsets of \mathcal{X} equipped with the usual union operator \cup is a semigroup. Such a semigroup might be used as a feature representation for complex objects by mapping an object to the set of its components, forgetting about the weights. The resulting representation would therefore be an element of $\mathcal{P}(\mathcal{X})$. A semigroup kernel k on $\mathcal{P}(\mathcal{X})$ measuring the similarity of two sets of points $A, B \in \mathcal{P}(\mathcal{X})$ would use the value taken by a given p.d. function φ on their union, namely $k(A, B) = \varphi(A \cup B)$. However we put aside this framework for two reasons. First, the union composition is idempotent (i.e., for all A in $\mathcal{P}(\mathcal{X})$, we have $A \cup A = A$) which as noted in Berg et al. (1984, Proposition 4.4.18) drastically restricts the class of possible p.d. functions. Second, such a framework defined by sets would ignore the frequency (or weights) of the components described in lists, which can be misleading when dealing with finite sets of components. Other problematic features would include the fact that $k(A, B)$ would be constant when $B \subset A$ regardless of its characteristics, and that comparing sets of very different sizes should be difficult.

In order to overcome these limitations we propose to represent a list of weighted points $z = (x_i, a_i)_{i=1}^d$, where for $1 \leq i \leq d$ we have $x_i \in \mathcal{X}$ and $a_i > 0$, by its image measure $\delta_z = \sum_{i=1}^d a_i \delta_{x_i}$, and

focus now on the Abelian semigroup $(M_+^b(\mathcal{X}), +)$ to define kernels between lists of weighted points. This representation is richer than the one suggested in the previous paragraph in the semigroup $(\mathcal{P}(\mathcal{X}), \cup)$ to consider the merger of two lists. First it performs the union of the supports; second the sum of such molecular measures also adds the weights of the points common to both measures, with a possible renormalization on those weights. Two important features of the original list are however lost in this mapping: the order of its elements and the original frequency of each element within the list as a weighted singleton. We assume for the rest of this paper that this information is secondary compared to the one contained in the image measure, namely its unordered support and the *overall* frequency of each point in that support. As a result, we study in the following sections p.d. functions on the semigroup $(M_+^b(\mathcal{X}), +)$, in particular on molecular measures, in order to define kernels on weighted lists of simple components.

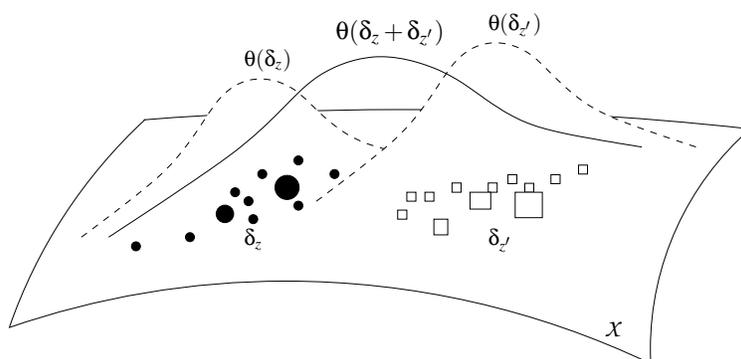


Figure 1: Measure representations of two lists z and z' . Each element of z (resp. z') list is represented by a black circle (resp. a white square), the size of which represents the associated weight. Five measures of interest are represented: the image measures δ_z and $\delta_{z'}$ of those weighted finite lists, the smoothed density estimates $\theta(\delta_z)$ and $\theta(\delta_{z'})$ of the two lists of points, and the smoothed density estimate $\theta(\delta_z + \delta_{z'})$ of the union of both lists.

Before starting the analysis of such p.d. functions, it should however be pointed out that several interesting semigroup p.d. kernels on measures are not directly applicable to molecular measures. For example, the first function we study below is only defined on the set of absolutely continuous measures with finite entropy. In order to overcome this limitation and be able to process complex objects in such situations, it is possible to think about alternative strategies to represent such objects by measures, as illustrated in Figure 1:

- The molecular measures δ_z and $\delta_{z'}$ as the image measures corresponding to the two weighted sets of points of z and z' , where dots and squares represent the different weights applied on each points;
- Alternatively, smoothed estimates of these distributions obtained for example by non-parametric or parametric statistical density estimation procedures, and represented by $\theta(\delta_z)$ and $\theta(\delta_{z'})$ in Figure 1. Such estimates can be considered if a p.d. kernel is only defined for absolutely continuous measures. When this mapping takes the form of estimation among a given family

of densities (through maximum likelihood for instance) this can also be seen as a prior belief assumed on the distribution of the objects;

- Finally, a smoothed estimate of the sum $\delta_z + \delta_{z'}$ corresponding to the merging of both lists, represented by $\theta(\delta_z + \delta_{z'})$, can be considered. Note that $\theta(\delta_z + \delta_{z'})$ might differ from $\theta(\delta_z) + \theta(\delta_{z'})$.

A kernel between two lists of points can therefore be derived from a p.d. function on $(M_+^b(\mathcal{X}), +)$ in at least three ways:

$$k(z, z') = \begin{cases} \varphi(\delta_z + \delta_{z'}), & \text{using } \varphi \text{ directly on molecular measures,} \\ \varphi(\theta(\delta_z) + \theta(\delta_{z'})), & \text{using } \varphi \text{ on smoothed versions of the molecular measures,} \\ \varphi(\theta(\delta_z + \delta_{z'})), & \text{evaluating } \varphi \text{ on a smoothed version of the sum.} \end{cases}$$

The positive definiteness of φ on $M_+^b(\mathcal{X})$ ensures positive definiteness of k only in the first two cases. The third expression can be seen as a special case of the first one, where we highlight the usage of a preliminary mapping on the sum of two measures; in that case $\varphi \circ \theta$ should in fact be p.d. on $(M_+^b(\mathcal{X}), +)$, or at least $(\text{Mol}_+(\mathcal{X}), +)$. Having defined the set of representations on which we will focus in this paper, namely measures on a set of components, we propose in the following section two particular cases of positive definite functions that can be computed through an addition between the considered measures. We then show how those quantities can be computed in the case of molecular measures in Section 4.

3. The Entropy and Inverse Generalized Variance Kernels

In this section we present two basic p.d. semigroup kernels for measures, motivated by a common intuition: the kernel between two measures should increase when the sum of the measures gets more “concentrated”. The two kernels differ in the way they quantify the concentration of a measure, using either its entropy or its variance. They are therefore limited to a subset of measures, namely the subset of measures with finite entropy and the subset of sub-probability measures with non-degenerated variance, but are extended to a broader class of measures, including molecular measures, in Section 4.

3.1 Entropy Kernel

We consider the subset of $M_+^b(\mathcal{X})$ of absolutely continuous measures with respect to the dominant measure ν , and identify in this section a measure with its corresponding density with respect to ν . We further limit the subset to the set of non-negative valued ν -measurable functions on \mathcal{X} with finite sum, such that

$$M_+^h(\mathcal{X}) \stackrel{\text{def}}{=} \{f : \mathcal{X} \rightarrow \mathbb{R}^+ \mid f \text{ is } \nu\text{-measurable, } |h(f)| < \infty, |f| < \infty\}$$

where we write for any measurable non-negative valued function g ,

$$h(g) \stackrel{\text{def}}{=} - \int_{\mathcal{X}} g \ln g \, d\nu,$$

(with $0 \ln 0 = 0$ by convention) and $|g| \stackrel{\text{def}}{=} \int_{\mathcal{X}} g \, d\nu$, consistently with the notation used for measures. If g is such that $|g| = 1$, $h(g)$ is its differential entropy. Using the following inequalities,

$$\begin{aligned} (a+b)\ln(a+b) &\leq a\ln a + b\ln b + (a+b)\ln 2, \text{ by convexity of } x \mapsto x \ln x, \\ (a+b)\ln(a+b) &\geq a\ln a + b\ln b, \end{aligned}$$

we have that $(M_+^h(\mathcal{X}), +)$ is an Abelian semigroup since for f, f' in $M_+^h(\mathcal{X})$ we have that $h(f+f')$ is bounded by integrating pointwise the inequalities above, the boundedness of $|f+f'|$ being also ensured. Following Rao (1987) we consider the quantity

$$J(f, f') \stackrel{\text{def}}{=} h\left(\frac{f+f'}{2}\right) - \frac{h(f) + h(f')}{2}, \quad (2)$$

known as the *Jensen divergence* (or Jensen-Shannon divergence) between f and f' , which as noted by Fuglede and Topsøe (2004) can be seen as a symmetrized version of the Kullback-Leibler (KL) divergence D , since

$$J(f, f') = \frac{1}{2}D\left(f \parallel \frac{f+f'}{2}\right) + \frac{1}{2}D\left(f' \parallel \frac{f+f'}{2}\right).$$

The expression of Equation (2) fits our framework of devising semigroup kernels, unlike the direct use of the KL divergence (Moreno et al., 2004) which is neither symmetric nor negative definite. As recently shown in Endres and Schindelin (2003) and Österreicher and Vajda (2003), \sqrt{J} is a metric on $M_+^h(\mathcal{X})$ which is a direct consequence of J 's negative definiteness proven below, through Berg et al. (1984, Proposition 3.3.2) for instance. The Jensen-Divergence was also recently reinterpreted as a special case of a wider family of metrics on $M_+^b(\mathcal{X})$ derived from a particular family of Hilbertian metrics on \mathbb{R}_+ as presented in Hein and Bousquet (2005). The comparison between two densities f, f' is in that case performed by integrating pointwise the squared distance $d^2(f(x), f'(x))$ between the two densities over \mathcal{X} , using for d a distance chosen among a suitable family of metrics in \mathbb{R}_+ to ensure that the final value is independent of the dominant measure ν . The considered family for d is described in Fuglede and Topsøe (2004) through two parameters, a family of which the Jensen Divergence is just a special case as detailed in Hein and Bousquet (2005). The latter work shares with this paper another similarity, which lies in the “kernelization” of such quantities defined on measures through a prior kernel on the space of components, as will be reviewed in Section 5. However, of all the Hilbertian metrics introduced in Hein and Bousquet (2005), the Jensen-Divergence is the only one that can be related to the semigroup framework used throughout this paper.

Note finally that a positive definite kernel k is said to be infinitely divisible if $-\ln k$ is a negative definite kernel. As a consequence, any positive exponentiation $k^\beta, \beta > 0$ of an infinitely divisible kernel is a positive definite kernel.

Proposition 1 *h is a negative definite function on the semigroup $M_+^h(\mathcal{X})$. As a consequence e^{-h} is a positive definite function on $M_+^h(\mathcal{X})$ and its normalized counterpart, $k_h \stackrel{\text{def}}{=} e^{-J}$ is an infinitely divisible positive definite kernel on $M_+^h(\mathcal{X}) \times M_+^h(\mathcal{X})$.*

Proof It is known that the real-valued function $r : y \mapsto -y \ln y$ is n.d. on \mathbb{R}_+ as a semigroup endowed with addition (Berg et al., 1984, Example 6.5.16). As a consequence the function $f \mapsto r \circ f$ is n.d. on $M_+^h(\mathcal{X})$ as a pointwise application of r since $r \circ f$ is integrable w.r.t ν . For any real-valued n.d. kernel k and any real-valued function g , we have trivially that $(y, y') \mapsto k(y, y') + g(y) + g(y')$

remains negative definite. This allows first to prove that $h(\frac{f+f'}{2})$ is also n.d. through the identity $h(\frac{f+f'}{2}) = \frac{1}{2}h(f+f') + \frac{\ln 2}{2}(|f| + |f'|)$. Subtracting the normalization factor $\frac{1}{2}(h(f) + h(f'))$ gives the negative definiteness of J . This finally yields the positive definiteness of k_h as the exponential of the negative of a n.d. function through Schoenberg's theorem (Berg et al., 1984, Theorem 3.2.2). ■

Note that only e^{-h} is a semigroup kernel strictly speaking, since e^{-J} involves a normalized sum (through the division by 2) which is not associative. While both e^{-h} and e^{-J} can be used in practice on non-normalized measures, we name more explicitly $k_h = e^{-J}$ the *entropy kernel*, because what it indeed quantifies when f and f' are normalized (i.e., such that $|f| = |f'| = 1$) is the difference of the average of the entropy of f and f' from the entropy of their average. The subset of absolutely continuous *probability* measures on (\mathcal{X}, ν) with finite entropies, namely $\{f \in M_+^h(\mathcal{X}), \text{ s.t. } |f| = 1\}$ is not a semigroup since it is not closed by addition, but we can nonetheless define the restriction of J and hence k_h on it to obtain a p.d. kernel on probability measures inspired by semigroup formalism.

3.2 Inverse Generalized Variance Kernel

We assume in this subsection that \mathcal{X} is an Euclidian space of dimension n endowed with Lebesgue's measure ν . Following the results obtained in the previous section, we propose under these restrictions a second semigroup p.d. kernel between measures which uses generalized variance. The generalized variance of a measure, namely the determinant of its variance matrix, is a quantity homogeneous to a volume in \mathcal{X} . This volume can be interpreted as a typical volume occupied by a measure when considering only its second order moments, making it hence a useful quantification of its dispersion. Besides being easy to compute in the case of molecular measures, this quantity is also linked to entropy if we consider that for normal laws $\mathcal{N}(m, \Sigma)$ the following relation holds:

$$\frac{1}{\sqrt{\det \Sigma}} \propto e^{-h(\mathcal{N}(m, \Sigma))}.$$

Through this observation, we note that considering the Inverse of the Generalized Variance (IGV) of a measure is equivalent to considering the value taken by e^{-2h} on its maximum likelihood normal law. We will put aside this interpretation in this section, before reviewing it with more care in Section 7.

Let us define the variance operator on measures μ with finite first and second moment of $M_+^b(\mathcal{X})$ as

$$\Sigma(\mu) \stackrel{\text{def}}{=} \mu[xx^\top] - \mu[x]\mu[x]^\top.$$

Note that $\Sigma(\mu)$ is always a positive semi-definite matrix when μ is a sub-probability measure, that is when $|\mu| \leq 1$, since

$$\Sigma(\mu) = \mu[(x - \mu[x])(x - \mu[x])^\top] + (1 - |\mu|)\mu[x]\mu[x]^\top.$$

We call $\det \Sigma(\mu)$ the generalized variance of a measure μ , and say a measure μ is *non-degenerated* if $\det \Sigma(\mu)$ is non-zero, meaning that $\Sigma(\mu)$ is of full rank. The subset of $M_+^b(\mathcal{X})$ of such measures with total weight equal to 1 is denoted by $M_+^v(\mathcal{X})$; $M_+^v(\mathcal{X})$ is convex through the following proposition:

Proposition 2 $M_+^v(\mathcal{X}) \stackrel{\text{def}}{=} \{\mu \in M_+^b(\mathcal{X}) : |\mu| = 1, \det \Sigma(\mu) > 0\}$ is a convex set, and more generally for $\lambda \in [0, 1)$, $\mu' \in M_+^b(\mathcal{X})$ such that $|\mu'| = 1$ and $\mu \in M_+^v(\mathcal{X})$, $(1 - \lambda)\mu + \lambda\mu' \in M_+^v(\mathcal{X})$.

Proof We use the following identity,

$$\Sigma((1-\lambda)\mu + \lambda\mu') = (1-\lambda)\Sigma(\mu) + \lambda\Sigma(\mu') + \lambda(1-\lambda)(\mu[x] - \mu'[x])(\mu[x] - \mu'[x])^\top,$$

to derive that $\Sigma((1-\lambda)\mu + \lambda\mu')$ is a (strictly) positive-definite matrix as the sum of two positive semi-definite matrices and a strictly positive definite matrix $\Sigma(\mu)$. ■

$M_+^v(\mathcal{X})$ is not a semigroup, since it is not closed under addition. However we will work in this case on the mean of two measures in the same way we used their standard addition in the semigroup framework of $M_+^b(\mathcal{X})$.

Proposition 3 *The real-valued kernel k_v defined on elements μ, μ' of $M_+^v(\mathcal{X})$ as*

$$k_v(\mu, \mu') = \frac{1}{\det \Sigma(\frac{\mu + \mu'}{2})}$$

is positive definite.

Proof Let y be an element of \mathcal{X} . For any $N \in \mathbb{N}$, any $c_1, \dots, c_N \in \mathbb{R}$ such that $\sum_i c_i = 0$ and any $\mu_1, \dots, \mu_N \in M_+^v(\mathcal{X})$ we have

$$\begin{aligned} \sum_{i,j} c_i c_j y^\top \Sigma\left(\frac{\mu_i + \mu_j}{2}\right) y &= \sum_{i,j} c_i c_j y^\top \left(\frac{1}{2} \mu_i [xx^\top] + \frac{1}{2} \mu_j [xx^\top] - \right. \\ &\quad \left. \frac{1}{4} \left(\mu_i [x] \mu_i [x]^\top + \mu_j [x] \mu_j [x]^\top + \mu_j [x] \mu_i [x]^\top + \mu_i [x] \mu_j [x]^\top \right) \right) y \\ &= -\frac{1}{4} \sum_{i,j} c_i c_j y^\top \left(\mu_j [x] \mu_i [x]^\top + \mu_i [x] \mu_j [x]^\top \right) y \\ &= -\frac{1}{2} \left(\sum_i c_i y^\top \mu_i [x] \right)^2 \leq 0, \end{aligned}$$

making thus the function $\mu, \mu' \mapsto y^\top \Sigma(\frac{\mu + \mu'}{2}) y$ negative-definite for any $y \in \mathcal{X}$. Using again Schoenberg's theorem (Berg et al., 1984, Theorem 3.2.2) we have that $\mu, \mu' \mapsto e^{-y^\top \Sigma(\frac{\mu + \mu'}{2}) y}$ is positive definite and so is the sum $\frac{1}{(2\pi)^{\frac{d}{2}}} \int_{\mathcal{X}} e^{-y^\top \Sigma(\frac{\mu + \mu'}{2}) y} \nu(dy)$ which is equal to $1/\sqrt{\det \Sigma(\frac{\mu + \mu'}{2})}$ ensuring thus the positive-definiteness of k_v as its square. ■

Both entropy and IGV kernels are defined on subsets of $M_+^b(\mathcal{X})$. Since we are most likely to use them on molecular measures or smooth measures (as discussed in Section 2.2), we present in the following section practical ways to apply them in that framework.

4. Semigroup Kernels on Molecular Measures

The two positive definite functions defined in Sections 3.1 and 3.2 cannot be applied in the general case to $\text{Mol}_+(\mathcal{X})$ which as exposed in Section 2 is our initial goal. In the case of the entropy kernel, molecular measures are generally not absolutely continuous with respect to ν (except on

finite spaces), and they have therefore no entropy; we solve this problem by mapping them into $M_+^h(\mathcal{X})$ through a smoothing kernel. In the case of the IGV, the estimates of variances might be poor if the number of points in the lists is not large enough compared to the dimension of the Euclidean space; we perform in that case a regularization by adding a unit-variance correlation matrix to the original variance. This regularization is particularly important to pave the way to the kernelized version of the IGV kernel presented in the next section, when \mathcal{X} is not Euclidian but simply endowed with a prior kernel κ .

The application of both the entropy kernel and the IGV kernel to molecular measures requires a previous renormalization to set the total mass of the measures to 1. This technical renormalization is also beneficial, since it allows a consistent comparison of two weighted lists even when their size and total mass is very different. All molecular measures in this section, and equivalently all admissible bases, will hence be supposed to be normalized such that their total weight is 1, and $\text{Mol}_+^1(\mathcal{X})$ denotes the subset of $\text{Mol}_+(\mathcal{X})$ of such measures.

4.1 Entropy Kernel on Smoothed Estimates

We first define the Parzen smoothing procedure which allows to map molecular measures onto measures with finite entropy:

Definition 4 *Let κ be a probability kernel on \mathcal{X} with finite entropy, i.e., a real-valued function defined on \mathcal{X}^2 such that for any $x \in \mathcal{X}$, $\kappa(x, \cdot) : y \mapsto \kappa(x, y)$ satisfies $\kappa(x, \cdot) \in M_+^h(\mathcal{X})$ and $|\kappa(x, \cdot)| = 1$. The κ -Parzen smoothed measure of μ is the probability measure whose density with respect to ν is $\theta_\kappa(\mu)$, where*

$$\begin{aligned} \theta_\kappa : \text{Mol}_+^1(\mathcal{X}) &\longrightarrow M_+^h(\mathcal{X}) \\ \mu &\mapsto \sum_{x \in \text{supp } \mu} \mu(x) \kappa(x, \cdot). \end{aligned}$$

Note that for any admissible base $(x_i, a_i)_{i=1}^d$ of μ we have that $\theta_\kappa(\mu) = \sum_{i=1}^d a_i \kappa(x_i, \cdot)$. Once this mapping is defined, we use the entropy kernel to propose the following kernel on two molecular measures μ and μ' ,

$$k_h^\kappa(\mu, \mu') = e^{-J(\theta_\kappa(\mu), \theta_\kappa(\mu'))}.$$

As an example, let \mathcal{X} be an Euclidian space of dimension n endowed with Lebesgue's measure, and κ the isotropic Gaussian RBF kernel on that space, namely

$$\kappa(x, y) = \frac{1}{(2\pi\sigma)^{\frac{n}{2}}} e^{-\frac{\|x-y\|^2}{2\sigma^2}}.$$

Given two weighted lists z and z' of components in \mathcal{X} , $\theta_\kappa(\delta_z)$ and $\theta_\kappa(\delta_{z'})$ are thus mixtures of Gaussian distributions on \mathcal{X} . The resulting kernel computes the entropy of $\theta_\kappa(\delta_z)$ and $\theta_\kappa(\delta_{z'})$ taken separately and compares it with that of their mean, providing a positive definite quantification of their overlap.

4.2 Regularized Inverse Generalized Variance of Molecular Measures

In the case of a molecular measure μ defined on an Euclidian space \mathcal{X} of dimension n , the variance $\Sigma(\mu)$ is simply the usual empirical estimate of the variance matrix expressed in an orthonormal basis

of \mathcal{X} :

$$\Sigma(\mu) = \mu[xx^\top] - \mu[x]\mu[x]^\top = \sum_{i=1}^d a_i x_i x_i^\top - \left(\sum_{i=1}^d a_i x_i \right) \left(\sum_{i=1}^d a_i x_i \right)^\top,$$

where we use an admissible base $\gamma = (x_i, a_i)_{i=1}^d$ of μ to give a matrix expression of $\Sigma(\mu)$, with all points x_i expressed as column vectors. Note that this matrix expression, as would be expected from a function defined on measures, does not depend on the chosen admissible base. Given such an admissible base, let $X_\gamma = [x_i]_{i=1..d}$ be the $n \times d$ matrix made of all column vectors x_i and Δ_γ the diagonal matrix of weights of γ taken in the same order $(a_i)_{1 \leq i \leq d}$. If we write I_d for the identity matrix of rank d and $\mathbb{1}_{d,d}$ for the $d \times d$ matrix composed of ones, we have for any base γ of μ that:

$$\Sigma(\mu) = X_\gamma(\Delta_\gamma - \Delta_\gamma \mathbb{1}_{d,d} \Delta_\gamma) X_\gamma^\top,$$

which can be rewritten as

$$\Sigma(\mu) = X_\gamma(I_d - \Delta_\gamma \mathbb{1}_{d,d}) \Delta_\gamma (I_d - \mathbb{1}_{d,d} \Delta_\gamma) X_\gamma^\top,$$

noting that $(\Delta_\gamma \mathbb{1}_{d,d})^2 = \Delta_\gamma \mathbb{1}_{d,d}$ since $\text{trace} \Delta_\gamma = 1$.

The determinant of $\Sigma(\mu)$ can be equal to zero when the size of the support of μ is smaller than n , the dimension of \mathcal{X} , or more generally when the linear span of the points in the support of μ does not cover the whole space \mathcal{X} . This problematic case is encountered in Section 5 when we consider kernelized versions of the IGV, using an embedding of \mathcal{X} into a functional Hilbert space of potentially infinite dimension. Mapping an element of $\text{Mol}_+^1(\mathcal{X})$ into $M_+^v(\mathcal{X})$ by adding to it any element of $M_+^v(\mathcal{X})$ through Proposition 2 would work as a regularization technique; for an arbitrary $\rho \in M_+^v(\mathcal{X})$ and a weight $\lambda \in [0, 1)$ we could use the kernel defined as

$$\mu, \mu' \mapsto \frac{1}{\det \Sigma \left(\lambda \frac{\mu + \mu'}{2} + (1 - \lambda) \rho \right)}.$$

We use in this section a different strategy inspired by previous works (Fukumizu et al., 2004, Bach and Jordan, 2002) further motivated in the case of covariance operators on infinite dimensional spaces as shown by Cuturi and Vert (2005). The considered regularization consists in modifying directly the matrix $\Sigma(\mu)$ by adding a small diagonal component ηI_n where $\eta > 0$ so that its spectrum never vanishes. When considering the determinant of such a regularized matrix $\Sigma(\mu) + \eta I_n$ this is equivalent to considering the determinant of $\frac{1}{\eta} \Sigma(\mu) + I_n$ up to a factor η^n , which will be a more suitable expression in practice. We thus introduce the regularized kernel k_v^η defined on measures $(\mu, \mu') \in M_+^b(\mathcal{X})$ with finite second moment as

$$k_v^\eta(\mu, \mu') \stackrel{\text{def}}{=} \frac{1}{\det \left(\frac{1}{\eta} \Sigma \left(\frac{\mu + \mu'}{2} \right) + I_n \right)}.$$

It is straightforward to prove that the regularized function k_v^η is a positive definite kernel on the measures of $M_+^b(\mathcal{X})$ with finite second-order moments using the same proof used in Proposition 3. If we now introduce

$$K_\gamma \stackrel{\text{def}}{=} \left[x_i^\top x_j \right]_{1 \leq i, j \leq d},$$

for the $d \times d$ matrix of dot-products associated with the elements of a base γ , and

$$\tilde{K}_\gamma \stackrel{\text{def}}{=} \left[(x_i - \sum_{k=1}^d a_k x_k)^\top (x_j - \sum_{k=1}^d a_k x_k) \right]_{1 \leq i, j \leq d} = (I_d - \mathbb{1}_{d,d} \Delta_\gamma) K_\gamma (I_d - \Delta_\gamma \mathbb{1}_{d,d}),$$

for its centered expression with respect to the mean of μ , we have the following result:

Proposition 5 *Let X be an Euclidian space of dimension n . For any $\mu \in \text{Mol}_+^1(X)$ and any admissible base γ of μ we have*

$$\det \left(\frac{1}{\eta} \tilde{K}_\gamma \Delta_\gamma + I_{l(\gamma)} \right) = \det \left(\frac{1}{\eta} \Sigma(\mu) + I_n \right).$$

Proof We omit the references to μ and γ in this proof to simplify matrix notations, and write $d = l(\gamma)$. Let \tilde{X} be the $n \times d$ matrix $[x_i - \sum_{j=1}^d a_j x_j]_{i=1..d}$ of centered column vectors enumerated in γ , namely $\tilde{X} = X(I_d - \Delta \mathbb{1}_{d,d})$. We have

$$\begin{aligned} \Sigma &= \tilde{X} \Delta \tilde{X}^\top, \\ \tilde{K} \Delta &= \tilde{X}^\top \tilde{X} \Delta. \end{aligned}$$

Through the singular value decomposition of $\tilde{X} \Delta^{\frac{1}{2}}$, it is straightforward to see that the non-zero elements of the spectrums of matrices $\tilde{K} \Delta, \Delta^{\frac{1}{2}} \tilde{X}^\top \tilde{X} \Delta^{\frac{1}{2}}$ and Σ are identical. Thus, regardless of the difference between n and d , we have

$$\det \left(\frac{1}{\eta} \tilde{K} \Delta + I_d \right) = \det \left(\frac{1}{\eta} \Delta^{\frac{1}{2}} \tilde{X}^\top \tilde{X} \Delta^{\frac{1}{2}} + I_d \right) = \det \left(\frac{1}{\eta} \tilde{X} \Delta \tilde{X}^\top + I_n \right) = \det \left(\frac{1}{\eta} \Sigma + I_n \right),$$

where the addition of identity matrices only introduces an offset of 1 for all eigenvalues. \blacksquare

Given two measures $\mu, \mu' \in \text{Mol}_+^1(X)$, the following theorem can be seen as a regularized equivalent of Proposition 3 through an application of Proposition 5 to $\mu'' = \frac{\mu + \mu'}{2}$.

Theorem 6 *Let X be an Euclidian space. The kernel k_v^η defined on two measures μ, μ' of $\text{Mol}_+^1(X)$ as*

$$k_v^\eta(\mu, \mu') = \frac{1}{\det \left(\frac{1}{\eta} \tilde{K}_\gamma \Delta_\gamma + I_{l(\gamma)} \right)},$$

where γ is any admissible base of $\frac{\mu + \mu'}{2}$, is p.d. and independent of the choice of γ .

Given two objects z, z' and their respective molecular measures δ_z and $\delta_{z'}$, the computation of the IGV for two such objects requires in practice an admissible base of $\frac{\delta_z + \delta_{z'}}{2}$ as seen in Theorem 6. This admissible base can be chosen to be of the cardinality of the support of the mixture of δ_z and $\delta_{z'}$, or alternatively be the simple merger of two admissible bases of z and z' with their weights divided by 2, without searching for overlapped points between both lists. This choice has no impact on the final value taken by the regularized IGV-kernel and can be arbitrated by computational considerations.

If we now take a practical look at the IGV's definition, we note that it can be applied but to cases where the component space X is Euclidian, and only if the studied measures can be summarized efficiently by their second order moments. These limitations do not seem very realistic in practice,

since \mathcal{X} may not have a vectorial structure, and the distribution of the components may not even be well represented by Gaussians in the Euclidian case. We propose to bypass this issue and introduce the usage of the IGV in a more flexible framework by using the kernel trick on the previous quantities, since the IGV of a measure can be expressed only through the dot-products between the elements of the support of the considered measure.

5. Inverse Generalized Variance on the RKHS Associated with a Kernel κ

As with many quantities defined by dot-products, one is tempted to replace the usual dot-product matrix \tilde{K} of Theorem 6 by an alternative Gram-matrix obtained through a p.d. kernel κ defined on \mathcal{X} . The advantage of such a substitution, which follows the well known “kernel trick” principle (Schölkopf and Smola, 2002), is multiple as it first enables us to use the IGV kernel on any non-vectorial space endowed with a kernel, thus in practice on any component space endowed with a kernel; second, it is also useful when \mathcal{X} is a dot-product space where a non-linear kernel can however be used (e.g., using Gaussian kernel) to incorporate into the IGV’s computation higher-order moment comparisons. We prove in this section that the inverse of the regularized generalized variance, computed in Proposition 5 through the centered dot-product matrix \tilde{K}_γ of elements of any admissible base γ of μ , is still a positive definite quantity if we replace \tilde{K}_γ by a centered Gram-matrix $\tilde{\mathcal{K}}_\gamma$, computed through an a priori kernel κ on \mathcal{X} , namely

$$\begin{aligned} \mathcal{K}_\gamma &= [\kappa(x_i, x_j)]_{1 \leq i, j \leq d} \\ \tilde{\mathcal{K}}_\gamma &= (I_d - \mathbb{1}_{d,d} \Delta_\gamma) \mathcal{K}_\gamma (I_d - \Delta_\gamma \mathbb{1}_{d,d}). \end{aligned}$$

This substitution follows also a general principle when considering kernels on measures. The “kernelization” of a given kernel defined on measures to take into account a prior similarity on the components, when computationally feasible, is likely to improve its overall performance in classification tasks, as observed in Kondor and Jebara (2003) but also in Hein and Bousquet (2005) under the “Structural Kernel” appellation. The following theorem proves that this substitution is valid in the case of the IGV.

Theorem 7 *Let \mathcal{X} be a set endowed with a p.d. kernel κ . The kernel*

$$k_\kappa^\eta(\mu, \mu') = \frac{1}{\det\left(\frac{1}{\eta} \tilde{\mathcal{K}}_\gamma \Delta_\gamma + I_{l(\gamma)}\right)}, \tag{3}$$

defined on two elements μ, μ' in $\text{Mol}_+^1(\mathcal{X})$ is positive definite, where γ is any admissible base of $\frac{\mu + \mu'}{2}$.

Proof Let $N \in \mathbb{N}$, $\mu_1, \dots, \mu_N \in \text{Mol}_+^1(\mathcal{X})$ and $(c_i)_{i=1}^N \in \mathbb{R}^N$. Let us now study the quantity $\sum_{i=1}^N c_i c_j k_\kappa^\eta(\mu_i, \mu_j)$. To do so we introduce by the Moore-Aronszajn theorem (Berlinet and Thomas-Agnan, 2003, p.19) the reproducing kernel Hilbert space Ξ with reproducing kernel κ indexed on \mathcal{X} . The usual mapping from \mathcal{X} to Ξ is denoted by ϕ , that is $\phi : \mathcal{X} \ni x \mapsto \kappa(x, \cdot)$. We define

$$\mathcal{Y} \stackrel{\text{def}}{=} \text{supp} \left(\sum_{i=1}^N \mu_i \right) \subset \mathcal{X},$$

the finite set which numbers all elements in the support of the N considered measures, and

$$\Upsilon \stackrel{\text{def}}{=} \text{span} \phi(\mathcal{Y}) \subset \Xi,$$

the linear span of the elements in the image of \mathcal{Y} through ϕ . Υ is a vector space whose finite dimension is upper-bounded by the cardinality of \mathcal{Y} . Endowed with the dot-product inherited from Ξ , we further have that Υ is Euclidian. Given a molecular measure $\mu \in \text{Mol}_+^1(\mathcal{Y})$, let $\phi(\mu)$ denote the image measure of μ in Υ , namely $\phi(\mu) = \sum_{x \in \mathcal{Y}} \mu(x) \delta_{\phi(x)}$. One can easily check that any admissible base $\gamma = (x_i, a_i)_{i=1}^d$ of μ can be used to provide an admissible base $\phi(\gamma) \stackrel{\text{def}}{=} (\phi(x_i), a_i)_{i=1}^d$ of $\phi(\mu)$. The weight matrices Δ_γ and $\Delta_{\phi(\gamma)}$ are identical and we further have $\tilde{\mathcal{K}}_\Upsilon = \tilde{\mathcal{K}}_{\phi(\gamma)}$ by the reproducing property, where $\tilde{\mathcal{K}}$ is defined by the dot-product of the Euclidian space Υ induced by κ . As a result, we have that $k_\kappa^\eta(\mu_i, \mu_j) = k_\nu^\eta(\phi(\mu_i), \phi(\mu_j))$ where k_ν^η is defined on $\text{Mol}_+^1(\Upsilon)$, ensuring the non-negativity

$$\sum_{i=1}^N c_i c_j k_\kappa^\eta(\mu_i, \mu_j) = \sum_{i=1}^N c_i c_j k_\nu^\eta(\phi(\mu_i), \phi(\mu_j)) \geq 0$$

and hence positive-definiteness of k_κ^η . ■

As bserved in the experimental section, the kernelized version of the IGV is more likely to be successful to solve practical tasks since it incorporates meaningful information on the components. Before observing these practical improvements, we provide a general study of the family of semigroup kernels on $M_+^b(\mathcal{X})$ by casting the theory of integral representations of positive definite functions on a semigroup (Berg et al., 1984) in the framework of measures, providing new results and possible interpretations of this class of kernels.

6. Integral Representation of Positive Definite Functions on a Set of Measures

In this section we study a general characterization of *all* p.d. functions on the whole semigroup $(M_+^b(\mathcal{X}), +)$, including thus measures which are not normalized. This characterization is based on a general integral representation theorem valid for any semigroup kernel, and is similar in spirit to the representation of p.d. functions obtained on Abelian groups through Bochner’s theorem (Rudin, 1962). Before stating the main results in this section we need to recall basic definitions of semicharacters and exponentially bounded function (Berg et al., 1984, chap. 4).

Definition 8 *A real-valued function ρ on an Abelian semigroup $(S, +)$ is called a semicharacter if it satisfies the following properties:*

- (i) $\rho(0) = 1$
- (ii) $\forall s, t \in S, \rho(s+t) = \rho(s)\rho(t)$.

It follows from the previous definition and the fact that $M_+^b(\mathcal{X})$ is 2-divisible (i.e., $\forall \mu \in M_+^b(\mathcal{X}), \exists \mu' \in M_+^b(\mathcal{X})$ s.t. $\mu = 2\mu'$) that semicharacters are nonnegative valued since it suffices to write that $\rho(\mu) = \rho(\frac{\mu}{2})^2$. Note also that semicharacters are trivially positive definite functions on S . We denote by S^* the set of semicharacters on $M_+^b(\mathcal{X})$, and by $\hat{S} \subset S^*$ the set of bounded semicharacters. S^* is a Hausdorff space when endowed with the topology inherited from \mathbb{R}^S having the topology of pointwise convergence. Therefore we can consider the set of Radon measures on S^* , namely $M_+^b(S^*)$.

Definition 9 *A function $f : M_+^b(\mathcal{X}) \rightarrow \mathbb{R}$ is called exponentially bounded if there exists a function $\alpha : M_+^b(\mathcal{X}) \rightarrow \mathbb{R}_+$ (called an absolute value) satisfying $\alpha(0) = 1$ and $\alpha(\mu + \mu') \leq \alpha(\mu)\alpha(\mu')$ for*

$\mu, \mu' \in M_+^b(\mathcal{X})$, and a constant $C > 0$ such that:

$$\forall \mu \in M_+^b(\mathcal{X}), \quad f(\mu) \leq C\alpha(\mu).$$

We can now state two general integral representation theorems for p.d. functions on semigroups (Berg et al., 1984, Theorems 4.2.5 and 4.2.8). These theorems being valid on any semigroup, they hold in particular on the particular semigroup $(M_+^b(\mathcal{X}), +)$.

Theorem 10 • A function $\varphi : M_+^b(\mathcal{X}) \rightarrow \mathbb{R}$ is p.d. and exponentially bounded if and only if it has an integral representation:

$$\varphi(s) = \int_{S^*} \rho(s) d\omega(\rho),$$

with $\omega \in M_+^c(S^*)$ (the set of Radon measures on S^* with compact support).

• A function $\varphi : M_+^b(\mathcal{X}) \rightarrow \mathbb{R}$ is p.d. and bounded if and only if it has an integral representation of the form:

$$\varphi(s) = \int_{\hat{S}} \rho(s) d\omega(\rho),$$

with $\omega \in M_+(\hat{S})$.

In both cases, if the integral representation exists, then there is uniqueness of the measure ω in $M_+(S^*)$.

In order to make these representations more constructive, we need to study the class of (bounded) semicharacters on $(M_+^b(\mathcal{X}), +)$. Even though we are not able to provide a complete characterization, even of bounded semicharacters, the following proposition introduces a large class of semicharacters, and completely characterizes the *continuous* semicharacters. For matters related to continuity of functions defined on $M_+^b(\mathcal{X})$, we will consider the weak topology of $M_+^b(\mathcal{X})$ which is defined in simple terms through the *portmanteau* theorem (Berg et al., 1984, Theorem 2.3.1). Note simply that if μ_n converges to μ in the weak topology then for any *bounded* measurable and continuous function f we have that $\mu_n[f] \rightarrow \mu[f]$. We further denote by $C(\mathcal{X})$ the set of continuous real-valued functions on \mathcal{X} and by $C^b(\mathcal{X})$ its subset of bounded functions. Both sets are endowed with the topology of pointwise convergence. For a function $f \in \mathbb{R}^{\mathcal{X}}$ we write ρ_f for the function $\mu \mapsto e^{\mu[f]}$ when the integral is well defined.

Proposition 11 A semicharacter $\rho : M_+^b(\mathcal{X}) \rightarrow \mathbb{R}$ is continuous on $(M_+^b(\mathcal{X}), +)$ endowed with the weak topology if and only if there exists $f \in C^b(\mathcal{X})$ such that $\rho = \rho_f$. In that case, ρ is a bounded semicharacter on $M_+^b(\mathcal{X})$ if and only if $f \leq 0$.

Proof For a continuous and bounded function f , the semicharacter ρ_f is well-defined. If a sequence μ_n in $M_+^b(\mathcal{X})$ converges to μ weakly, we have $\mu_n[f] \rightarrow \mu[f]$, which implies the continuity of ρ_f . Conversely, suppose ρ is weakly continuous. Define $f : \mathcal{X} \rightarrow [-\infty, \infty)$ by $f(x) = \log \rho(\delta_x)$. If a sequence x_n converges to x in \mathcal{X} , obviously we have $\delta_{x_n} \rightarrow \delta_x$ in the weak topology, and

$$\rho(\delta_{x_n}) \rightarrow \rho(\delta_x),$$

which means the continuity of f . To see the boundedness of f , assume the contrary. Then, we can find $x_n \in X$ such that either of $0 < f(x_n) \rightarrow \infty$ or $0 > f(x_n) \rightarrow -\infty$ holds. Let $\beta_n = |f(x_n)|$. Because the measure $\frac{1}{\beta_n} \delta_{x_n}$ converges weakly to zero, the continuity of ρ means

$$\rho\left(\frac{1}{\beta_n} \delta_{x_n}\right) \rightarrow 1,$$

which contradicts with the fact $\rho\left(\frac{1}{\beta_n} \delta_{x_n}\right) = e^{\frac{1}{\beta_n} f(x_n)} = e^{\pm 1}$. Thus, ρ_f is well-defined, weakly continuous on $M_+^b(X)$ and equal to ρ on the set of molecular measures. It is further equal to ρ on $M_+^b(X)$ through the denseness of molecular measures in $M_+^b(X)$, both in the weak and the pointwise topology (Berg et al., 1984, Proposition 3.3.5). Finally suppose now that ρ_f is bounded and that there exists x in X such that $f(x) > 0$. By $\rho_f(n\delta_x) = e^{nf(x)}$ which diverges with n we see a contradiction. The converse is straightforward. ■

Let ω be a bounded nonnegative Radon measure on the Hausdorff space of continuous real-valued functions on X , namely $\omega \in M_+^b(C(X))$. Given such a measure, we first define the subset M_ω of $M_+^b(X)$ as

$$M_\omega = \left\{ \mu \in M_+^b(X) \mid \sup_{f \in \text{supp } \omega} \mu[f] < +\infty \right\}.$$

M_ω contains the null measure and is a semigroup.

Corollary 12 *For any bounded Radon measure $\omega \in M_+^b(C(X))$, the following function φ is a p.d. function on the semigroup $(M_\omega, +)$:*

$$\varphi(\mu) = \int_{C(X)} \rho_f(\mu) d\omega(f). \tag{4}$$

If $\text{supp } \omega \subset C^b(X)$ then φ is continuous on M_ω endowed with the topology of weak convergence.

Proof For $f \in \text{supp } \omega$, ρ_f is a well defined semicharacter on M_ω and hence positive definite. Since

$$\varphi(\mu) \leq |\omega| \sup_{f \in \text{supp } \omega} \mu[f]$$

is bounded, φ is well defined and hence positive definite. Suppose now that $\text{supp } \omega \subset C^b(X)$ and let μ_n be a sequence of M_ω converging weakly to μ . By the bounded convergence theorem and continuity of all considered semicharacters (since all considered functions f are bounded) we have that:

$$\lim_{n \rightarrow \infty} \varphi(\mu_n) = \int_{C(X)} \lim_{n \rightarrow \infty} \rho_f(\mu_n) d\omega(f) = \varphi(\mu).$$

and hence φ is continuous w.r.t the weak topology. ■

When the measure ω is chosen in such a way that the integral (4) is tractable or can be approximated, then a valid p.d. kernel for measures is obtained; an example involving mixtures over exponential families is provided in Section 7.

Before exploiting this constructive representation, a few remarks should be pointed out. When using non-bounded functions (as is the case when using expectation or second-order moments of measures) the continuity of the integral φ is left undetermined to our knowledge, even when its existence is ensured. However, when X is compact we have that $C(X) = C^b(X)$ and hence continuity

on M_ω of any function φ constructed through corollary 12. Conversely, there exist continuous p.d. functions on $(M_+^b(\mathcal{X}), +)$ that can not be represented in the form (4). Although any continuous p.d. function can necessarily be represented as an integral of semicharacters by Theorem 10, the semicharacters involved in the representation are not necessarily continuous as in (4). An example of such a continuous p.d. function written as an integral of non-continuous semicharacters is exposed in Appendix A. It is an open problem to our knowledge to fully characterize continuous p.d. functions on $(M_+^b(\mathcal{X}), +)$.

7. Projection on Exponential Families through Laplace's Approximation

The constructive approach presented in corollary 12 can be used in practice to define kernels by restricting the space $C(\mathcal{X})$ to subspaces where computations are tractable. A natural way to do so is to consider a vector space of finite dimension s of $C(\mathcal{X})$, namely the span of a free family of s non-constant functions f_1, \dots, f_s of $C(\mathcal{X})$, and define a measure on that subspace by applying a measure on the weights associated with each function. The previous integral representation (4) would then take the form:

$$\varphi(\mu) = \int_{\Theta} e^{\mu[\sum_{i=1}^s \theta_i f_i]} \omega(d\theta),$$

where ω is now a bounded measure on a compact subset $\Theta \subseteq \mathbb{R}^s$ and μ is such that $\mu[f_i] < +\infty$ for $1 \leq i \leq s$. The subspace of $C(\mathcal{X})$ considered in this section is however slightly different, in order to take advantage of the natural benefits of exponential densities generated by all functions f_1, \dots, f_s . Following Amari and Nagaoka (2001, p.69), this requires the definition of the cumulant generating function of ν with respect to f_1, \dots, f_s as

$$\psi(\theta) \stackrel{\text{def}}{=} \log \nu[e^{\sum_{i=1}^s \theta_i f_i}],$$

such that for each $\theta \in \Theta$,

$$p_\theta \stackrel{\text{def}}{=} \exp\left(\sum_{i=1}^s \theta_i f_i - \psi(\theta)\right) \nu,$$

is a probability density, which defines an exponential family of densities on \mathcal{X} as θ varies in Θ . Rather than the direct span of functions f_1, \dots, f_s on Θ , this is equivalent to considering the hyper-surface $\{\sum_{i=1}^s \theta_i f_i - \psi(\theta)\}$ in $\text{span}\{f_1, \dots, f_s, -1\}$. This yields the following expression:

$$\varphi(\mu) = \int_{\Theta} e^{\mu[\sum_{i=1}^s \theta_i f_i - \psi(\theta)]} \omega(d\theta).$$

Following the notations of Amari and Nagaoka (2001) the η -parameters (or expectation parameters) of μ are defined as

$$\hat{\eta}_i \stackrel{\text{def}}{=} \frac{1}{|\mu|} \mu[f_i], \quad 1 \leq i \leq s,$$

and $\hat{\theta}$ stands for the θ -parameters of $\hat{\eta}$. We assume in the following approximations that $\hat{\theta} \in \Theta$ and recall two identities (Amari and Nagaoka, 2001, Chapters 3.5 & 3.6):

$$\chi(\theta) \stackrel{\text{def}}{=} \sum_{i=1}^s \theta_i \eta_i - \psi(\theta) = -h(\theta), \quad \text{the dual potential,}$$

$$D(\theta||\theta') = \psi(\theta) + \chi(\theta') - \sum_{i=1}^s \theta_i \eta'_i, \quad \text{the KL divergence,}$$

where we used the abbreviations $h(\theta) = h(p_\theta)$ and $D(\theta||\theta') = D(p_\theta||p_{\theta'})$. We can then write

$$\begin{aligned} \mu\left[\sum_{i=1}^s \theta_i f_i - \psi(\theta)\right] &= |\mu| \left(\sum_{i=1}^s \theta_i \hat{\eta}_i - \psi(\theta) \right) \\ &= |\mu| \left(\sum_{i=1}^s \hat{\theta}_i \hat{\eta}_i - \psi(\hat{\theta}) + \sum_{i=1}^s (\theta_i - \hat{\theta}_i) \hat{\eta}_i + \psi(\hat{\theta}) - \psi(\theta) \right) \\ &= -|\mu| (h(\hat{\theta}) + D(\hat{\theta}||\theta)), \end{aligned}$$

to obtain the following factorized expression,

$$\varphi(\mu) = e^{-|\mu|h(\hat{\theta})} \int_{\Theta} e^{-|\mu|D(\hat{\theta}||\theta)} \omega(d\theta). \tag{5}$$

The quantity $e^{-|\mu|h(\hat{\theta})}$ was already evoked in Section 3.2 when multivariate normal distributions were used to express the IGV kernel. When \mathcal{X} is an Euclidian space of dimension n , this is indeed equivalent to defining $s = n + n(n + 1)/2$ base functions, more precisely $f_i = x_i$ and $f_{ij} = x_i x_j$, and dropping the integral of Equation (5). Note that such functions are not bounded and that M_ω corresponds here to the set of measures with finite first and second order moments.

The integral of Equation (5) cannot be computed in a general case. The use of conjugate priors can however yield exact calculations, such as in the setting proposed by Cuturi and Vert (2005). In their work \mathcal{X} is a finite set of short sequences formed over an alphabet, functions f_i are all possible indicator functions of \mathcal{X} and ω is an additive mixture of Dirichlet priors. The kernel value is computed through a factorization inspired by the context-tree weighting algorithm (Willems et al., 1995). In the general case a numerical approximation can also be derived using Laplace’s method (Dieudonné, 1968) under the assumption that $|\mu|$ is large enough. To do so, first notice that

$$\begin{aligned} \frac{\partial D(\hat{\theta}||\theta)}{\partial \theta_i} \Big|_{\theta=\hat{\theta}} &= \frac{\partial \psi}{\partial \theta_i} \Big|_{\theta=\hat{\theta}} - \hat{\eta}_i = 0, \\ \frac{\partial D(\hat{\theta}||\theta)}{\partial \theta_i \partial \theta_j} &= \frac{\partial \psi}{\partial \theta_i \partial \theta_j} = g_{ij}(\theta), \end{aligned}$$

where $G_\theta = [g_{ij}(\theta)]$ is the Fisher information matrix computed in θ and hence a p.d. matrix. The following approximation then holds:

$$\varphi(\mu) \underset{|\mu| \rightarrow \infty}{\sim} e^{-|\mu|h(\hat{\theta})} \int_{\mathbb{R}^s} \omega(\hat{\theta}) e^{-\frac{|\mu|}{2}(\theta-\hat{\theta})^\top G_{\hat{\theta}}(\theta-\hat{\theta})} d\theta = e^{-|\mu|h(\hat{\theta})} \left(\frac{2\pi}{|\mu|} \right)^{\frac{s}{2}} \frac{\omega(\hat{\theta})}{\sqrt{\det G_{\hat{\theta}}}}$$

which can be simplified by choosing ω to be Jeffrey’s prior (Amari and Nagaoka, 2001, p.44), namely

$$\omega(d\theta) = \frac{1}{V} \sqrt{\det G_\theta} d\theta, \quad \text{where } V = \int_{\Theta} \sqrt{\det G_\theta} d\theta.$$

Up to a multiplication by V this provides an approximation of φ by $\tilde{\varphi}$ as

$$\varphi(\mu) \underset{|\mu| \rightarrow \infty}{\sim} \tilde{\varphi}(\mu) \stackrel{\text{def}}{=} e^{-|\mu|h(\hat{\theta})} \left(\frac{2\pi}{|\mu|} \right)^{\frac{s}{2}}.$$

The η -coordinates of μ are independent of the total weight $|\mu|$, hence $\tilde{\varphi}(2\mu) = \tilde{\varphi}(\mu)^2 \left(\frac{|\mu|}{4\pi}\right)^{\frac{s}{2}}$. This identity can be used to propose a renormalized kernel for two measures as

$$k(\mu, \mu') \stackrel{\text{def}}{=} \frac{\tilde{\varphi}(\mu + \mu')}{\sqrt{\tilde{\varphi}(2\mu)\tilde{\varphi}(2\mu')}} = \frac{e^{-(|\mu+\mu'|)h(p_{\mu+\mu'})}}{e^{-|\mu|h(p_\mu)-|\mu'|h(p_{\mu'})}} \left(\frac{2\sqrt{|\mu||\mu'|}}{|\mu| + |\mu'|} \right)^{\frac{s}{2}}.$$

where p_μ stands for $p_{\hat{\delta}_\mu}$. When μ and μ' are normalized such that their total weight coincides and is equal to β , we have that

$$k(\mu, \mu') = e^{-2\beta \left(h(p_{\mu'}) - \frac{h(p_\mu) + h(p_{\mu'})}{2} \right)}, \quad (6)$$

where $\mu'' = \mu + \mu'$. From Equation (6), we see that β can be tuned in practice and thought of as a width parameter. It should be large enough to ensure the consistency of Laplace's approximation and thus positive definiteness, while not too large at the same time to avoid diagonal dominance issues. In the case of the IGV kernel this tradeoff can however be put aside since the inverse of the IGV is directly p.d. as was proved in Proposition 3. However and to our knowledge this assertion does not stand in a more general case when the functions f_1, \dots, f_s are freely chosen.

8. Experiments on Images of the MNIST Database

We present in this section experimental results and discussions on practical implementations of the IGV kernels on a benchmark experiment of handwritten digits classification. We focus more specifically on the kernelized version of the IGV and discuss its performance with respect to other kernels. The entropy kernel performed very poorly in the series of experiments presented here, besides requiring a time consuming Monte Carlo computation, which is why we do not consider it in this section. We believe however that in more favourable cases, notably when the considered measures are multinomials, the entropy kernel and its structural variants (Hein and Bousquet, 2005) may provide good results.

8.1 Linear IGV Kernel

Following the previous work of Kondor and Jebara (2003), we have conducted experiments on 500 and 1000 images (28×28 pixels) taken from the MNIST database of handwritten digits (black shapes on a white background), with 50 (resp. 100) images for each digit. To each image z we randomly associate a set of d distinct points which are black (intensity superior to 190) in the image. In this case the set of components is $\{1, \dots, 28\} \times \{1, \dots, 28\}$ which we map onto points with coordinates between 0 and 1, thus defining $\mathcal{X} = [0, 1]^2$. The linear IGV kernel as described in Section 3.2 is equivalent to using the linear kernel $\kappa((x_1, y_1), (x_2, y_2)) = x_1x_2 + y_1y_2$ on a non-regularized version of the kernelized-IGV. It also boils down to fitting Gaussian bivariate-laws on the points and measuring the similarity of two measures by performing variance estimation on the samples taken first separately and then together. The resulting variances can be diagonalized to obtain three diagonal variance matrices, which can be seen as performing PCA on the sample,

$$\Sigma(\mu) = \begin{pmatrix} \Sigma_{1,1} & 0 \\ 0 & \Sigma_{2,2} \end{pmatrix}, \quad \Sigma(\mu') = \begin{pmatrix} \Sigma'_{1,1} & 0 \\ 0 & \Sigma'_{2,2} \end{pmatrix}, \quad \Sigma(\mu'') = \begin{pmatrix} \Sigma''_{1,1} & 0 \\ 0 & \Sigma''_{2,2} \end{pmatrix}.$$

and the value of the kernel is computed through

$$k_v(\mu, \mu') = \frac{\sqrt{\Sigma_{1,1}\Sigma_{2,2}\Sigma'_{1,1}\Sigma'_{2,2}}}{\Sigma''_{1,1}\Sigma''_{2,2}}.$$

This ratio is for instance equal to 0.3820 for two handwritten digits in the case shown in Figure 2. The linear IGV manages a good discrimination between ones and zeros. Indeed, ones are shaped

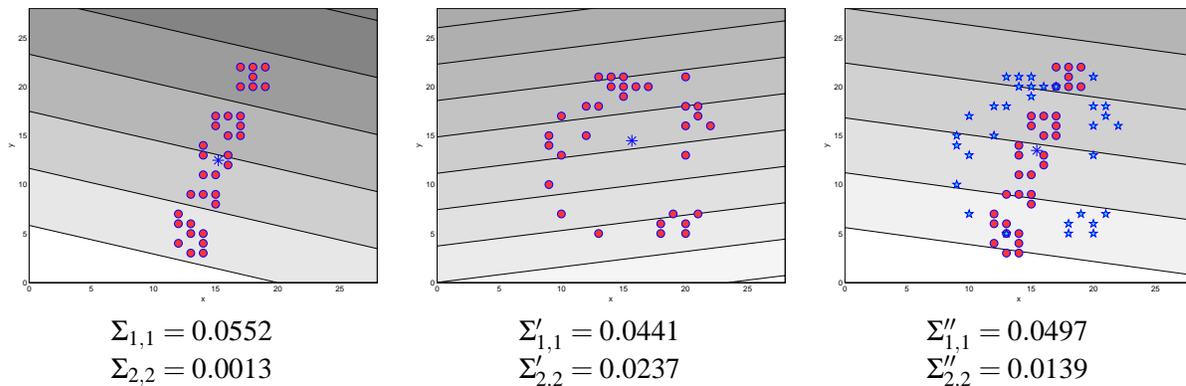


Figure 2: Weighted PCA of two different measures and their mean, with their first principal component shown. Below are the variances captured by the first and second principal components, the generalized variance being the product of those two values.

as sticks, and hence usually have a strong variance carried by their first component, followed by a weak second component. On the other hand, the variance of zeros is more equally distributed between the first and second axes. When both weighted sets of points are united, the variance of the mean of both measures has an intermediary behaviour in that respect, and this suffices to discriminate numerically both images. However this strategy fails when using numbers which are not so clearly distinct in shape, or more precisely whose surface cannot be efficiently expressed in terms of Gaussian ellipsoids. To illustrate this we show in Figure 3 the Gram matrix of the linear IGV on 60 images, namely 20 zeros, 20 ones and 20 twos. Though images of ones can be efficiently discriminated from the two other digits, we clearly see that this is not the case between zeros and twos, whose support may seem similar if we try to capture them through Gaussian laws. In practice, the results obtained with the linear IGV on this particular task were so unadapted to the learning goal that the SVM's trained based on that methodology did not converge in most cases, which is why we discarded it.

8.2 Kernelized IGV

Following previous works (Kondor and Jebara, 2003, Wolf and Shashua, 2003) and as suggested in the initial discussion of Section 5, we use in this section a Gaussian kernel of width σ to incorporate a prior knowledge on the pixels, and equivalently to define the reproducing kernel Hilbert space Ξ by using

$$\kappa((x_1, y_1), (x_2, y_2)) = e^{-\frac{(x_1-x_2)^2+(y_1-y_2)^2}{2\sigma^2}}.$$

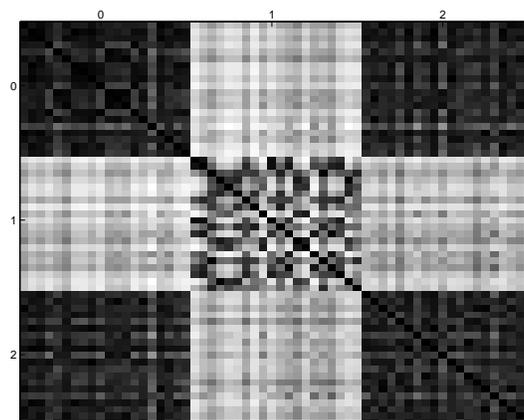


Figure 3: Normalized Gram matrix computed with the linear IGV kernel of twenty images of “0”, “1” and “2” displayed in that order. Darker spots mean values closer to 1, showing that the restriction to “0” and “1” yields good separation results, while “0” and “2” can hardly be discriminated using variance analysis.

As pointed out by Kondor and Jebara (2003), the pixels are no longer seen as points but rather as functions (Gaussian bells) defined on the components space $[0, 1]^2$. To illustrate this approach we show in Figure 4 the first four eigenfunctions of three measures μ_1 , μ_0 and $\frac{\mu_1 + \mu_0}{2}$ built from the image of a handwritten “1” and “0” with their corresponding eigenvalues, as well as for images of “2” and “0” in Figure 5.

Setting σ , the width of κ , to define the functions contained in the RKHS Ξ is not enough to fully characterize the values taken by the kernelized IGV. We further need to define η , the regularization parameter, to control the weight assigned to smaller eigenvalues in the spectrum of Gram matrices. Both parameters are strongly related, since the value of σ controls the range of the typical eigenvalues found in the spectrum of Gram matrices of admissible bases, whereas η acts as a scaling parameter for those eigenvalues as can be seen in Equation (3). Indeed, using a very small σ value, which means Ξ is only defined by peaked Gaussian bells around each pixels, yields diagonally dominant Gram matrices very close to the identity matrix. The resulting eigenvalues for $\tilde{\mathcal{K}}\Delta$ are then all very close to $\frac{1}{d}$, the inverse of the amount of considered points. On the contrary, a large value for σ yields higher values for the kernel, since all points would be similar to each other and Gram matrices would turn close to the matrix $\mathbb{1}_{d,d}$ with a single significant eigenvalue and all others close to zero. We address these issues and study the robustness of the final output of the k-IGV kernel in terms of classification error by doing preliminary experiments where both η and σ vary freely.

8.3 Experiments on the SVM Generalization Error

To study the behaviour and the robustness of the IGV kernel under different parameter settings, we used two ranges of values for η and σ :

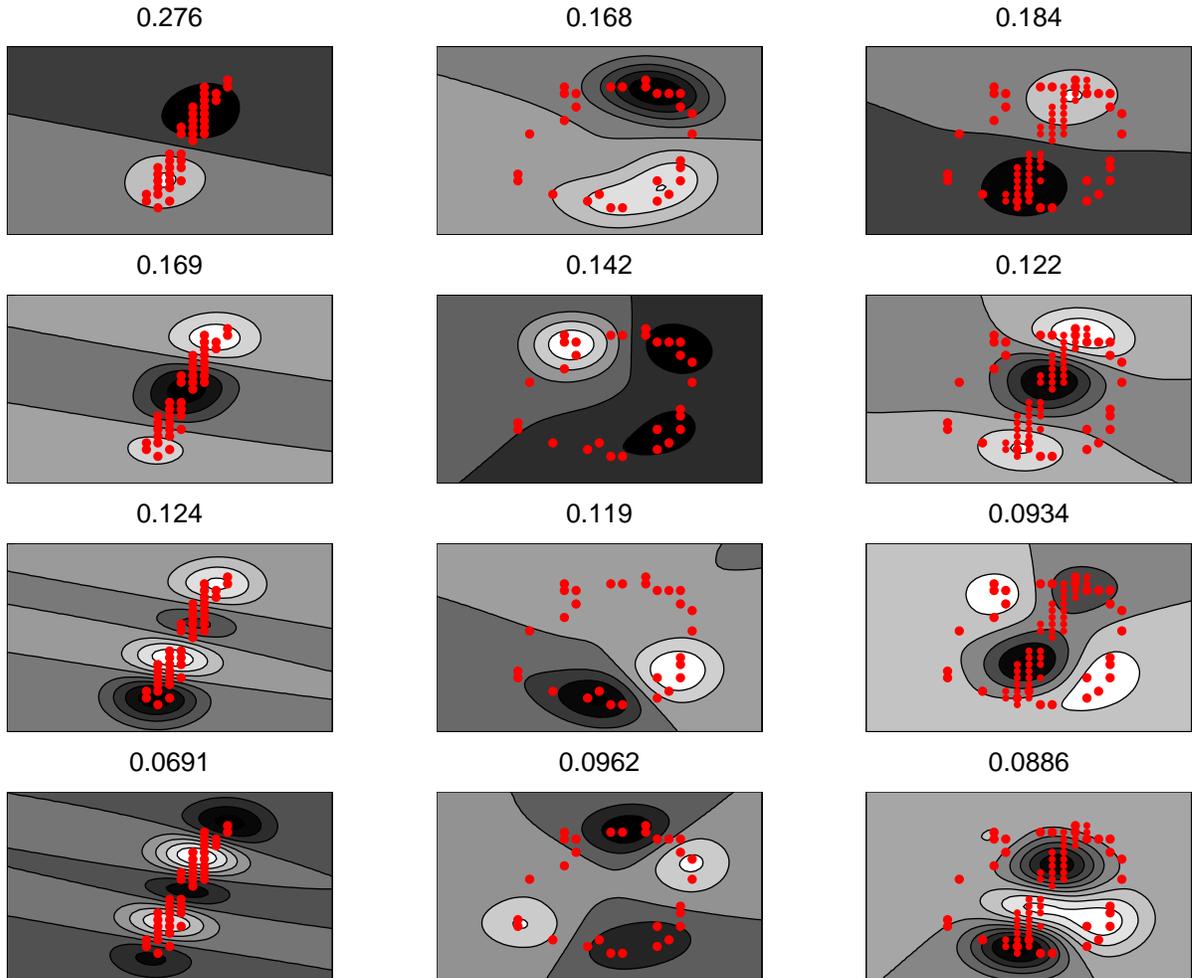


Figure 4: The four first eigenfunctions of respectively three empirical measures μ_1 (first column), μ_0 (second column) and $\frac{\mu_1 + \mu_0}{2}$ (third column), displayed with their corresponding eigenvalues, using $\eta = 0.01$ and $\sigma = 0.1$.

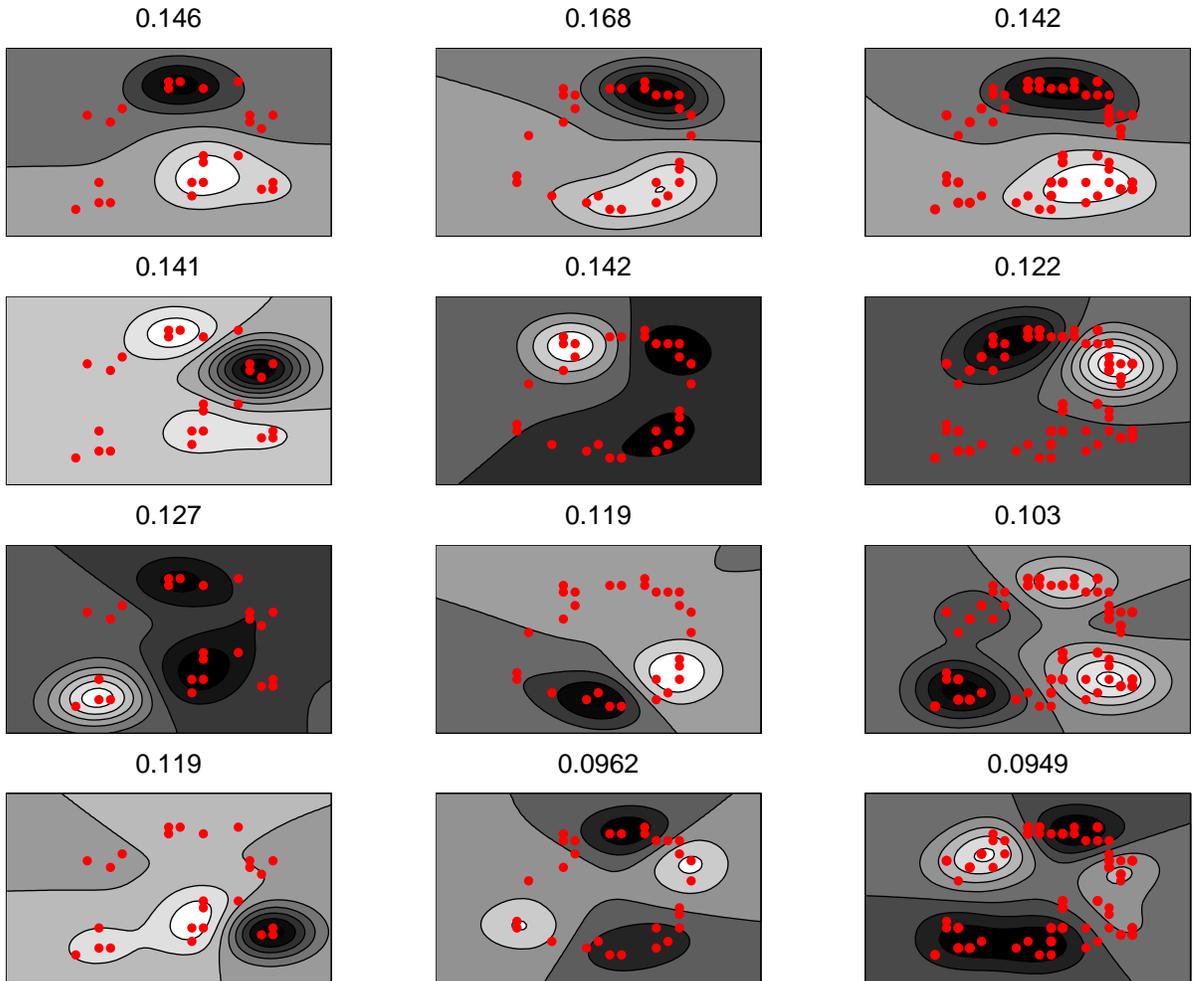


Figure 5: Same representation as in Figure 4, with μ_2 , μ_0 and $\frac{\mu_2 + \mu_0}{2}$.

$$\eta \in 10^{-2} \times \{0.1, 0.3, 0.5, 0.8, 1, 1.5, 2, 3, 5, 8, 10, 20\}$$

$$\sigma \in \{0.05, 0.1, 0.12, 0.15, 0.18, 0.20, 0.25, 0.3\}.$$

For each kernel k_{κ}^{η} defined by a (σ, η) couple, we trained 10 binary SVM classifiers (each one trained to recognize each digit versus all other digits) on a training fold of our 500 images dataset such that the proportion of each class was kept to be one tenth of the total size of the training set. Using then the test fold, our decision for each submitted image was determined by the highest SVM score proposed by the 10 trained binary SVM's. To determine train and test points, we led a 3-fold cross validation, namely randomly splitting our total dataset into 3 balanced subsets, using successively 2 subsets for training and the remaining one for testing (that is roughly 332 images for training and 168 for testing). The test error was not only averaged on those cross-validations folds but also on 5 different fold divisions. All the SVM experiments in this experimental section were run using the spider¹ toolbox. Most results shown here did not improve by choosing different soft margin C parameters, we hence just set $C = \infty$ as suggested by default by the authors of the toolbox.

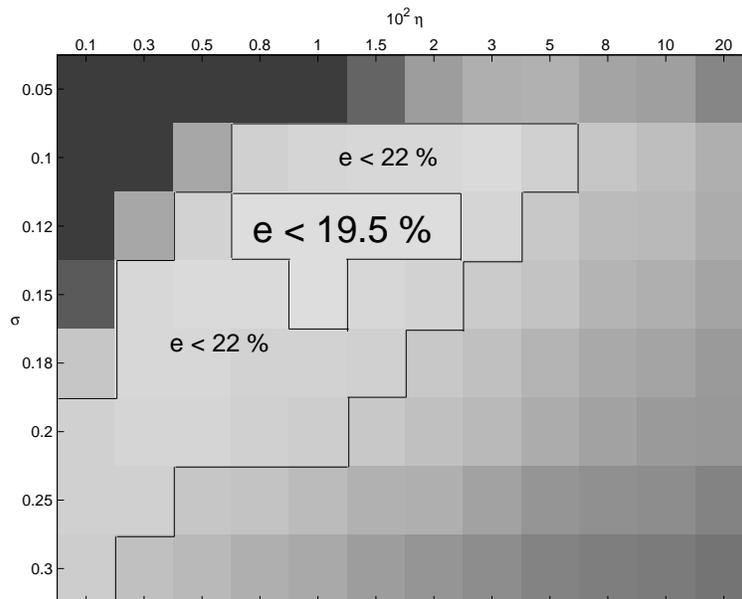


Figure 6: Average test error (displayed as a grey level) of different SVM handwritten character recognition experiments using 500 images from the MNIST database (each seen as a set of 25 to 30 randomly selected black pixels), carried out with 3-fold (2 for training, 1 for test) cross validations with 5 repeats, where parameters η (regularization) and σ (width of the Gaussian kernel) have been tuned to different values.

The error rates are graphically displayed in Figure 6 using a grey-scale plot. Note that for this benchmark the best testing errors were reached using a σ value of 0.12 with an η parameter within 0.008 and 0.02, this error being roughly 19.5%. All values below and on the right side of this zone

1. see <http://www.kyb.tuebingen.mpg.de/bs/people/spider/>

are below 32.5%, which is the value reached on the lower right corner. All standard deviations with respect to multiple cross-validations of those results were inferior to 2.3%, the whole region under 22% being under a standard deviation of 1%. Those preliminary tests show that the IGV kernel has an overall robust performance within what could be considered as a sound range of values for both η and σ . Note that the optimal range of parameter found in this experiment only applies to the specific sampling procedure that was used in this case (25 to 30 points), and may not be optimal for larger matrices. However the stability observed here led us to discarding further tuning of σ and η when the amount of sampled points is different. We simply applied $\sigma = 0.1$ and $\eta = 0.01$ for the remaining of the experimental section.

As in Kondor and Jebara (2003), we also compared the results obtained with the k-IGV to the standard RBF kernel performed on the images seen as binary vectors of $\{0, 1\}^{28 \times 28}$ further normalized so that their components sums up to 1. Using the same range for σ that was previously tested, we applied the formula $k(z, z') = e^{-\frac{\|z-z'\|^2}{2\sigma^2}}$. Since the RBF kernel is grounded on the exact overlapping between two images we expect it to perform poorly with few pixels and significantly better when d grows, while we expect the k-IGV to capture more quickly the structure of the images with fewer pixels through the kernel κ . This is illustrated in Figure 7 where the k-IGV outperforms significantly the RBF kernel, reaching with a sample of less than 30 points a performance the RBF kernel only reaches above 100 points. Taking roughly all black points in the images, by setting $d = 200$ for instance, the RBF kernel error is still 17.5%, an error the IGV kernel reaches with roughly 35 points.

Finally, we compared the kernelized-version of the Bhattacharyya kernel (k-B) proposed in Kondor and Jebara (2003), the k-IGV, the polynomial kernel and the RBF kernel by using a larger database of the first 1,000 images in MNIST (100 images for each of the 10 digits), selecting randomly $d = 40, 50, 60, 70$ and 80 points and performing the cross-validation methodology previously detailed. The polynomial kernel was performed seeing the images as binary vectors of $\{0, 1\}^{28 \times 28}$ and applying the formula $k_{b,d}(z, z') = (z \cdot z' + b)^d$. We followed the observations of Kondor and Jebara (2003) concerning parameter tuning for the k-B kernel but found out that it performed better using the same set of parameters used for the k-IGV. The results presented in Table 1 of the k-IGV kernel show a consistent improvement over all other kernels for this benchmark of 1000 images, under all sampling schemes.

We did not use the kernel described by Wolf and Shashua (2003) in our experiments because of its poor scaling properties for a large amount of considered points. Indeed, the kernel proposed by Wolf and Shashua (2003) takes the form of the product of d cosines values where d is the cardinality of the considered sets of points, thus yielding negligible values in practice when d is large as in our case. Their SVM experiments were limited to 6 or 7 points while we mostly consider lists of more than 40 points here. This problem of poor scaling which in practice produces a diagonal-dominant kernel led us to discarding this method in our comparison. All semigroup kernels presented in this paper are grounded on statistical estimation, which makes their values stable under variable sizes of samples through renormalization, a property shared with the work of Kondor and Jebara (2003). Beyond a minimal amount of points needed to perform sound estimation, the size of submitted samples influences positively the accuracy of the k-IGV kernel. A large sample size can lead however to computational problems since the value of the k-IGV-kernel requires not only the computation of the centered Gram-matrix \mathcal{K} and a few matrix multiplications, but also the computation of a determinant, an operation which can quickly become prohibitive since it has a complexity of $O(d^{2.3})$ where d is the size of the considered Gram matrix. Although we did not opti-

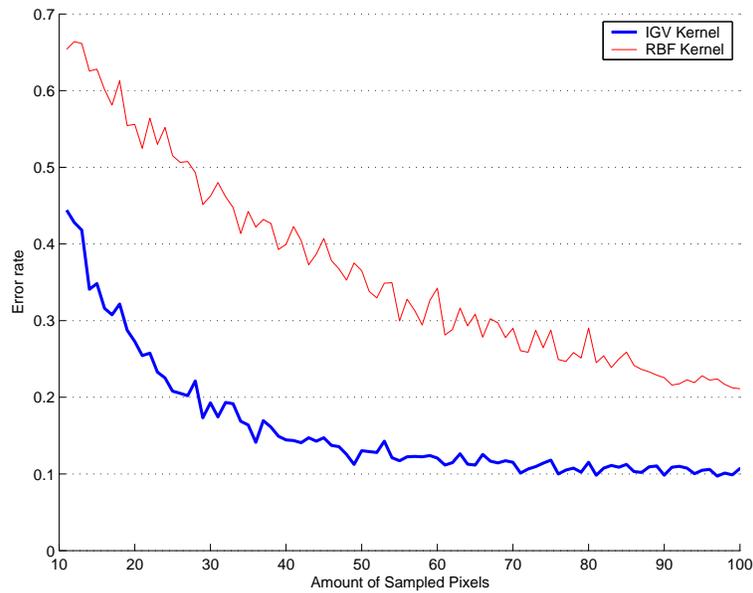


Figure 7: Average test error with RBF ($\sigma = 0.2$) and k-IGV ($\sigma = 0.1$ and $\eta = 0.01$) kernels led on 90 different samplings of 500 images. The curves show an overall trend that both kernels perform better when they are given more points to compute the similarity between two images. If we consider $d = 200$, the RBF kernel error is 0.175, that is 17.5%, a threshold the IGV kernel reaches with slightly more than 35 points. Each sampling corresponds to a different amount of sampled points d , those samplings being ordered increasingly with d . Each sampling has been performed independently which explains the bumpiness of those curves.

mize the computations of both k-B and k-IGV kernels (by storing precomputed values for instance or using numerical approximations in the computation of the determinant), this computational cost in the case of a naive implementation, illustrated by the running times displayed in Table 1, remains an issue that needs to be addressed in practical applications.

Sample Size	Gaussian $\sigma = 0.1$	Polynomial $b = 10; d = 4$	k-B $\eta = 0.01; \sigma = 0.1$	k-IGV $\eta = 0.01; \sigma = 0.1$
40 pixels	32.2 (1)	31.3 (1.5)	19.1 (1500)	16.2 (1000)
50 "	28.5 (1)	26.3 (1.5)	17.1 (2500)	14.7 (1400)
60 "	24.5 (1)	22.0 (1.5)	15.8 (3600)	14.6 (2400)
70 "	22.2 (1)	19.5 (1.5)	15.1 (4100)	13.1 (2500)
80 "	20.3 (1)	17.4 (1.5)	14.5 (5500)	12.8 (3200)

Table 1: SVM Error rate in percents of different kernels used on a benchmark test of recognizing digits images, where only 40 to 80 black points were sampled from the original images. The 1,000 images were randomly split into 3 balanced sets to perform cross validation (2 for training and 1 for testing), the error being first averaged over 5 such splits, the whole process being repeated again over 3 different random samples of points. Running times are indicated in minutes.

9. Conclusion

We presented in this work a new family of kernels between measures. Such kernels are defined through prior functions which should ideally quantify the concentration of a measure. Once such a function is properly defined, the kernel computation goes through the evaluation of the function on the two measures to be compared and on their mixture. As expected when dealing with concentration of measures, two intuitive tools grounded on information theory and probability, namely entropy and variance, prove to be useful to define such functions. Their expression is however still complex in terms of computational complexity, notably for the k-IGV kernel. Computational improvements or numerical simplifications should be brought forward to ensure a feasible implementation for large-scale tasks involving tens of thousands of objects.

An attempt to define and understand the general structure of p.d. functions on measures was also presented, through a representation as integrals of elementary functions known as semicharacters. We are investigating further theoretical properties and characterizations of both semicharacters and positive definite functions on measures. The choice of alternative priors on semicharacters to propose other meaningful kernels, with convenient properties on molecular measures for instance, is also a subject of future research. As for practical applications, these kernels can be naturally applied on complex objects seen as molecular measures. We also expect to perform further experiments to measure the performance of semigroup kernels on a diversified sample of challenging tasks, including cases where the space of components is not a vector space, notably when the considered measures are multinomials on a finite component space endowed with a kernel.

Acknowledgments

The authors would like to thank Francis Bach and Jérémie Jakubowicz for fruitful discussions, Imre Risi Kondor for sharing his code with us, anonymous reviewers for their comments which improved the quality of the paper and Xavier Dupré for his help on the MNIST database. MC acknowledges a JSPS doctoral short-term grant which made his stay in Japan possible. KF was supported by JSPS KAKENHI 15700241 and a research grant from the Inamori Foundation. JPV is supported by NHGRI NIH award R33 HG003070 and by the ACI “Nouvelles Interfaces des Mathématiques” of the French Ministry of Research. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors’ views.

Appendix A : an Example of Continuous Positive Definite Function Given by Noncontinuous Semicharacters

Let \mathcal{X} be the unit interval $[0, 1]$ hereafter. For any t in \mathcal{X} , a semicharacter on $M_+^b(\mathcal{X})$ is defined by

$$\rho_{h_t}(\mu) = e^{\mu([0,t])},$$

where $h_t(x) = I_{[0,t]}(x)$ is the index function of the interval $[0, t]$. Note that ρ_{h_t} is *not* continuous for $t \in [0, 1)$ by Proposition 11.

For $\mu \in M_+^b(\mathcal{X})$, the function $t \mapsto \mu([0, t])$ is bounded and non-decreasing, thus, Borel-measurable, since the discontinuous points are countable at most. A positive definite function on $M_+^b(\mathcal{X})$ is defined by

$$\varphi(\mu) = \int_0^1 \rho_{h_t}(\mu) dt.$$

This function is continuous, while it is given by the integral of noncontinuous semicharacters.

Proposition *The positive definite function φ is continuous and exponentially bounded.*

Proof Suppose μ_n converges to μ weakly in $M_+^b(\mathcal{X})$. We write $F_n(t) = \mu_n([0, t])$ and $F(t) = \mu([0, t])$. Because μ_n and μ are finite measures, the weak convergence means

$$F_n(t) \rightarrow F(t)$$

for any continuous point of F . Since the set of discontinuous points of F is at most countable, the above convergence holds almost everywhere on X with Lebesgue measure. From the weak convergence, we have $F_n(1) \rightarrow F(1)$, which means there exists $M > 0$ such that $\sup_{t \in \mathcal{X}, n \in \mathbb{N}} F_n(t) < M$. By the bounded convergence theorem, we obtain

$$\lim_{n \rightarrow \infty} \varphi(\mu_n) = \lim_{n \rightarrow \infty} \int_0^1 e^{F_n(t)} dt = \int_0^1 e^{F(t)} dt = \varphi(\mu).$$

For the exponential boundedness, by taking an absolute value $\alpha(\mu) = e^{\mu(X)}$, we have

$$|\varphi(\mu)| \leq \int_0^1 \alpha(\mu) dt = \alpha(\mu).$$

■

References

- Shotaro Akaho. A kernel method for canonical correlation analysis. In *Proceedings of International Meeting on Psychometric Society (IMPS2001)*, 2001.
- Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*. AMS vol. 191, 2001.
- Francis Bach and Michael Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.
- Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag, 1984.
- Alain Berlinet and Christine Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, 2003.
- B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.
- Marco Cuturi and Jean-Philippe Vert. The context-tree kernel for strings. *Neural Networks*, 2005. In press.
- Marco Cuturi and Jean-Philippe Vert. Semigroup kernels on finite sets. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 329–336. MIT Press, Cambridge, MA, 2005.
- Jean Dieudonné. *Calcul Infinitésimal*. Hermann, Paris, 1968.
- Dominik M. Endres and Johannes E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, 2003.
- Bent Fuglede and Flemming Topsøe. Jensen-shannon divergence and hilbert space embedding. In *Proc. of the Internat. Symposium on Information Theory*, page 31, 2004.
- Kenji Fukumizu, Francis Bach, and Michael Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.
- M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. January 2005.
- Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.
- Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms*. Kluwer Academic Publishers, Dordrecht, 2002. ISBN 0-7923-7679-X.
- Risi Kondor and Tony Jebara. A kernel between sets of vectors. In *Proceedings of the International Conference on Machine Learning*, 2003.

- John Lafferty and Guy Lebanon. Information diffusion kernels. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
- Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing 2002*, pages 564–575. World Scientific, 2002.
- Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for svm protein classification. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.
- Thomas Melzer, Michael Reiter, and Horst Bischof. Nonlinear feature extraction using generalized canonical correlation analysis. In *Proceedings of International Conference on Artificial Neural Networks (ICANN)*, pages 353–360, 2001.
- Pedro J. Moreno, Purdy P. Ho, and Nuno Vasconcelos. A kullback-leibler divergence based kernel for svm classification in multimedia applications. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
- Ferdinand Österreicher and Igor Vajda. A new class of metric divergences on probability spaces and its applicability in statistics. *Annals of the Institute of Statistical Mathematics*, 55:639–653, 2003.
- C. R. Rao. Differential metrics in probability spaces. In S.-I. Amari, O.E. Barndorff-Nielsen, R.E. Kass, S.L. Lauritzen, and C.R. Rao, editors, *Differential Geometry in Statistical Inference*, Hayward, CA, 1987. Institute of Mathematical Statistics.
- Walter Rudin. *Fourier Analysis on Groups*. John Wiley & sons, 1962.
- Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.
- Matthias Seeger. Covariance kernels from bayesian generative models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 905–912, Cambridge, MA, 2002. MIT Press.
- F. M. J. Willems, Y. M. Shtarkov, and Tj. J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, pages 653–664, 1995.
- Lior Wolf and Amnon Shashua. Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, 4:913–931, 2003.

Separating a Real-Life Nonlinear Image Mixture

Luís B. Almeida

LUIS.ALMEIDA@LX.IT.PT

*Instituto de Telecomunicações
Instituto Superior Técnico
Av. Rovisco Pais, 1
1049-00 Lisboa, Portugal*

Editor: Aapo Hyvärinen

Abstract

When acquiring an image of a paper document, the image printed on the back page sometimes shows through. The mixture of the front- and back-page images thus obtained is markedly nonlinear, and thus constitutes a good real-life test case for nonlinear blind source separation.

This paper addresses a difficult version of this problem, corresponding to the use of “onion skin” paper, which results in a relatively strong nonlinearity of the mixture, which becomes close to singular in the lighter regions of the images. The separation is achieved through the MISEP technique, which is an extension of the well known INFOMAX method. The separation results are assessed with objective quality measures. They show an improvement over the results obtained with linear separation, but have room for further improvement.

Keywords: ICA, blind source separation, nonlinear mixtures, nonlinear separation, image mixture, image separation

1. Introduction

When an image of a paper document is acquired, e.g. through scanning, photographing or photocopying, the image printed on the back page sometimes shows through. This is normally due to partial transparency of the paper, and results in the acquisition of a mixture of the images from the front and back pages. It is usually possible to obtain two different mixtures, by acquiring both sides of the document. This is a situation that seems suited for handling by blind source separation (BSS) techniques. The main difficulty is that the images that are acquired are nonlinear mixtures of the original images printed on each of the sides of the paper. This is, therefore, an interesting test case for nonlinear BSS methods, with potential application in scanners, photocopiers and in document processing in general.

This paper addresses a difficult instance of this problem, in which the paper that is used is of the “onion skin” type. This creates a mixture that has a relatively strong nonlinearity, and that is close to singular in the lighter parts of the images. For separation we use MISEP, which is a nonlinear independent component analysis (ICA) technique (Almeida, 2003b). MISEP is a generalization of the well known INFOMAX technique of linear ICA (Bell and Sejnowski, 1995), extending it in two directions: (1) being able to handle nonlinear mixtures, and (2) using output nonlinearities that adapt to the statistical distributions of the extracted components.

Besides the separation itself, an important practical issue in this specific situation is the alignment of the two mixture images. One might think that, by an appropriate translation and rotation, the images from the two sides of the document could be brought into good alignment with each other. It was found, however, that scanners normally introduce slight geometrical distortions that make it necessary to use local alignment techniques to obtain an image alignment that is adequate for separation. That alignment issue is also addressed in this paper, because it is an important step of the image processing that needs to be done.

Published results concerning nonlinear BSS in real-life problems are still very few. To the author's knowledge, and apart from an earlier version of the present work (Almeida and Faria, 2004), the only published report of blind source separation of a real-life nonlinear mixture in which the recovery of the original sources can be confirmed is (Haritopoulos et al., 2002). Some other applications of nonlinear ICA to real-life data, e.g. (Lappalainen and Honkela, 2000; Lee and Batzoglou, 2003), do not provide means to confirm whether real sources were recovered.

This manuscript's structure is as follows: Section 2 provides a brief overview of nonlinear separation methods. Section 3 presents a short summary of the MISEP method, to outline its basic principles and to set the notation. Section 4 describes the experimental conditions, including image printing, acquisition and alignment. Section 5 presents the experimental results, which are assessed with objective measures of separation quality. Section 6 concludes.

In the printed version of this paper some of the details of some images may be lost due to the printing process. However, the paper is freely available online, and in the electronic online version one can zoom in on the images (scatter plots and images of sources, mixtures and separated components) to better view the details. In the pdf version (~ 7 MB) the images are encoded in JPEG format and therefore show some artifacts, which become noticeable on close inspection. The postscript version shows the images without artifacts, but corresponds to a larger file (~ 14 MB). The two versions are available at

<http://www.lx.it.pt/~lbalmeida/papers/AlmeidaJMLR05.pdf>, and
<http://www.lx.it.pt/~lbalmeida/papers/AlmeidaJMLR05.ps.zip>.

The source and mixture images used in this paper are available online at
<http://www.lx.it.pt/~lbalmeida/ica/seethrough>.
 The separation routines that were used to produce the results are available at
<http://www.lx.it.pt/~lbalmeida/ica/seethrough/code/jmlr05>.

2. Overview of Nonlinear ICA Methods

In this section we provide a short overview of some of the main nonlinear ICA methods. This overview is necessarily very brief, and the reader is referred to an overview paper (Jutten and Karhunen, 2004) for more complete information.

It is interesting to note that one of the very early works on ICA (Schmidhuber, 1992) already proposed a nonlinear method. Although being based on an interesting principle (minimization of predictability of each extracted component by the other components) it was rather impractical and computationally heavy.

The essential uniqueness of the solution of linear ICA (Comon, 1994), together with the greater simplicity of linear separation and with the fact that many naturally occurring mixtures are essentially linear, led to a quick development of linear ICA. The work on nonlinear ICA probably was

slowed down mostly by its inherent ill-posedness and by its greater complexity, but development of nonlinear methods has continued steadily (e.g. Burel, 1992; Deco and Brauer, 1995; Marques and Almeida, 1999; Palmieri et al., 1999; Theis et al., 2003). The methods that have received the strongest attention in recent years are very briefly outlined in the next paragraphs.

Ensemble learning (Lappalainen and Honkela, 2000) is a Bayesian method and, as such, uses prior distributions as a form of regularization, to handle the ill-posedness problem. It is computationally heavy, but has produced some interesting results, including an extension to the separation of nonlinearly mixed dynamical processes (Valpola and Karhunen, 2002).

Kernel-based nonlinear ICA (Harmeling et al., 2003) essentially consists of linear ICA performed on a high-dimensional space that is a nonlinear transformation of the original space of mixture observations. In the form in which it was presented in the cited reference, it used the temporal structure of the signals to perform the linear ICA operation. This apparently helped it to effectively deal with the ill-posedness problem, and allowed it to yield some impressive results on artificial, strongly nonlinear mixtures. The method seems to be quite tractable, in computational terms.

MISEP (Almeida, 2003b) is an extension of INFOMAX (Bell and Sejnowski, 1995) into the nonlinear domain. It uses regularization to deal with the ill-posedness problem, and is computationally tractable. It is described in more detail in the next section, since it is the method used in the present paper.

A special class of methods that deserves mention deals with nonlinear mixtures which are constrained so as to make the result of ICA essentially unique, as in linear ICA. The most representative class corresponds to the so-called post-nonlinear (PNL) mixtures (Taleb and Jutten, 1999). These are linear mixtures followed by component-wise invertible nonlinearities. The interest of this class resides both in its unique separability and in the fact that it corresponds to well identified practical situations: linear mixtures observed by nonlinear sensors. PNL mixtures and their extensions have had a considerable development (see Jutten and Karhunen, 2004, for references).

3. Overview of the MISEP Method

MISEP (Almeida, 2003b) is a generalization of the INFOMAX method of linear ICA (Bell and Sejnowski, 1995). We recall that the latter method, although initially introduced under a principle of maximum information preservation, was later shown to be interpretable as a maximum likelihood method (Pearlmutter and Parra, 1996), and also as a method based on the minimization of the mutual information (MI) of the extracted components (Hyvärinen and Oja, 2000). We briefly recall the latter interpretation, albeit using a reasoning different from the one given in that reference.

If \mathbf{Y} is a vector with random components Y_i , we define the mutual information of the components of \mathbf{Y} as

$$I(\mathbf{Y}) = \sum_i H(Y_i) - H(\mathbf{Y}) \quad (1)$$

where, for continuous variables, as is the case here, H denotes Shannon's differential entropy

$$H(X) = - \int p(x) \log p(x) dx. \quad (2)$$

In this equation $p(x)$ is the probability density of the scalar random variable X (we denote probability density functions by $p(\cdot)$, the function's argument clarifying which random variable is being considered; this is a slight abuse of notation, but helps to keep expressions simpler and does not create any confusion). A similar definition holds for $H(\mathbf{X})$, where \mathbf{X} is a random vector, the difference

being that the random variable is now multidimensional and the integral in (2) becomes a multiple integral, encompassing the whole domain of \mathbf{X} .

Mutual information is a good measure of statistical dependence. $I(\mathbf{Y})$ measures the amount of information that is shared among the random variables Y_i . It is always positive, except if these variables are mutually statistically independent, in which case it is zero. $I(\mathbf{Y})$ is also equal to the Kullback-Leibler divergence between the product of the marginal densities, $\prod_i p(y_i)$ and the true joint density, $p(\mathbf{y})$. These two densities are equal if and only if the components Y_i are mutually independent.

Minimization of the mutual information of the extracted components is therefore a good criterion for independent component analysis. An interesting and useful property of mutual information, that we shall use ahead, is that if we apply invertible, possibly nonlinear, transformations to the random variables, $Z_i = \psi_i(Y_i)$, the mutual information doesn't change: $I(\mathbf{Z}) = I(\mathbf{Y})$.

INFOMAX uses a network with the structure depicted in Fig. 1. Block \mathbf{F} performs the separation proper, the separated components being y_i . \mathbf{F} is linear, corresponding just to a product by a matrix. The blocks ψ_i are auxiliary, being used only during the training phase. Each of these blocks performs an invertible, increasing transformation $z_i = \psi_i(y_i)$, whose counter-domain is the interval $(0, 1)$.

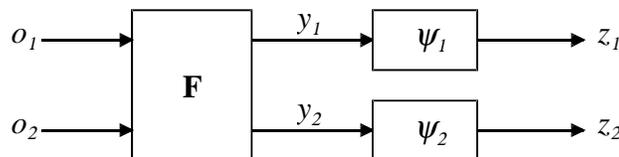


Figure 1: Network structure used in INFOMAX and in MISEP. In INFOMAX, \mathbf{F} is an adaptive linear block, and the ψ_i are fixed a priori. In MISEP, \mathbf{F} can be nonlinear, and both \mathbf{F} and ψ_i are adaptive.

If we choose each ψ_i as the cumulative distribution function (CDF) of the corresponding Y_i , it is easy to see that each of the Z_i will be uniformly distributed in $(0, 1)$, resulting in $p(z_i) = 1$ for z_i in that interval, and $H(Z_i) = 0$. Therefore,

$$\begin{aligned}
 I(\mathbf{Y}) &= I(\mathbf{Z}) \\
 &= \sum_i H(Z_i) - H(\mathbf{Z}) \\
 &= -H(\mathbf{Z}).
 \end{aligned} \tag{3}$$

Mutual information is hard to minimize directly, but (3) shows that, under the stated conditions, this minimization is equivalent to the maximization of the output entropy $H(\mathbf{Z})$, a maximization which is much easier to achieve. INFOMAX works by optimizing \mathbf{F} such that $H(\mathbf{Z})$ is maximized. We won't go into the details here, but the reader can consult (Bell and Sejnowski, 1995) or (Hyvärinen and Oja, 2000) for a deeper discussion.

As said above, MISEP extends INFOMAX in two directions. The first is being able to deal with nonlinear mixtures. This is achieved by allowing block \mathbf{F} , in Fig. 1, to be nonlinear. We have often implemented this block by means of a multilayer perceptron (MLP), but essentially any adaptive

nonlinear structure can be used. For example, a radial basis function network has been used in (Almeida, 2003a), and a specialized structure in (Almeida and Faria, 2004).

The second direction in which MISEP extends INFOMAX, is by making the output transformations ψ_i adaptive. As we have seen above, each ψ_i should correspond to the CDF of the corresponding extracted source, for the maximization of the output entropy to correspond to the minimization of the mutual information of the extracted components. The a-priori choice of the ψ_i functions in INFOMAX can be seen as a user-made, prior assumption about the distributions of the sources. In MISEP the ψ_i blocks are adaptive, being implemented by means of adequately constrained MLPs. It can be shown that maximization of the output entropy $H(\mathbf{Z})$ leads each of these blocks to estimate the corresponding CDF, while simultaneously leading \mathbf{F} to minimize the mutual information $I(\mathbf{Y})$ (Almeida, 2003b). Therefore, maximizing the output entropy simultaneously adapts the ψ_i blocks and leads to the minimization of the mutual information $I(\mathbf{Y})$.

An issue that has frequently been discussed is whether nonlinear blind source separation, based on ICA, is feasible in practice. This debate has to do with the fact that nonlinear ICA, with no additional constraints, is an ill-posed problem, having an infinite number of solutions that are not related to one another in any simple way (Darmois, 1953; Hyvärinen and Pajunen, 1999; Marques and Almeida, 1999). Therefore we cannot expect that, just by extracting independent components, one will be able to recover the original sources that were nonlinearly mixed. This is to be contrasted with the situation in linear ICA/BSS in which, under very mild constraints, there exists essentially only one solution (Comon, 1994). In linear ICA, if independent components are extracted, they must correspond to the original sources, apart from possible scaling and permutation. This author has argued that in the nonlinear case, when the mixture is not too strongly nonlinear, adequate regularization should allow the handling of the ill-posedness of nonlinear ICA, still allowing the approximate recovery of the sources. The nonlinearities considered in this paper would be classified by the author as of “medium intensity”. As we shall see below, approximate source recovery was possible, and the indetermination of nonlinear ICA didn’t lead to inadequate separation.

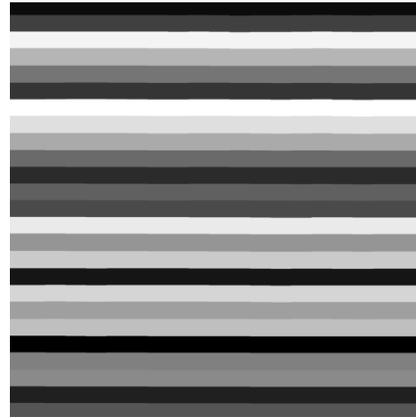
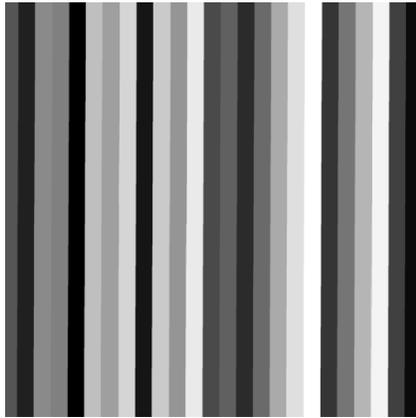
4. Experimental Setup

In this section we describe the experimental setup, including details of image printing, acquisition and preprocessing

4.1 Source Images

We used five image mixtures as test cases. The corresponding pairs of source images are shown in Figs. 2 and 3. The main properties of these image pairs are as follows:

1. In the first pair, each image consists of 25 uniform bars with intensities that are uniformly spaced between black and white, and are randomly ordered. The first image has vertical bars, and the second image is just the first one rotated by 90° . Thus, by construction, the intensities of the two images are independent, and each of the images has an intensity distribution which is close to uniform.
2. The second pair consists of images of natural scenes with a relatively high degree of variability and relatively small details. This causes a strong “mixing” of intensities, and the two sources are approximately independent from each other. However, the small details tend to make image superposition (due to imperfect separation) hard to notice visually.



In this example we are creating mixtures that involve natural images, printed text and graphs. The special characteristic of printed text and graphs is that they normally involve just two intensity levels (black and white) although, due to the above mentioned noise, these will appear, in the scanned images, as two clusters of intensity levels.

The separation of mixtures of two-level images, such as printed text, may be much easier than the separation of grayscale images. In fact, at least in the case of mixtures that are not too strong, a simple thresholding procedure may yield the desired results. Such a procedure can be easily performed by hand with most image processing programs, and should not be hard to automate. In such a case the use of more general blind source separation methods might be an overkill, both because it would involve a much larger amount of processing and because it might actually yield worse results. This is an extreme case in which prior knowledge about the sources can strongly simplify the separation process.

In the case of grayscale mixtures, the use of a separation method based on a good model of the physical mixing process should yield much better results than the use of a generic nonlinear separation method. A physical model could have a small number of parameters to be estimated, and would thus allow a much more precise estimation. Furthermore, it might avoid the inherent ill-posedness of nonlinear blind separation, which is currently addressed through regularization. The parameters of such a model could be estimated by an independent component analysis criterion.

Another issue of interest is the definition of separation criteria that are more suited for images or for printed documents than statistical independence. In fact, images and/or text from the opposite pages of a printed document can easily happen not to be independent from one other. For examples, images of landscapes tend to be lighter on the top than on the bottom, inducing a correlation between intensities of both. Also, in printed text with regularly spaced lines, the lines from both sides of the paper may happen to fall on top of each other, or the lines from one side may fall on the intervals of the lines from the other side, also inducing a significant correlation between intensities from both sides of the document. It would be interesting to use criteria based on a notion of image complexity, but these may not be easy to define, and may be even harder to use as criteria for optimizing a source separation system.



Figure 2: The first three pairs of source images, before printing. The images have been cropped, and one image in each pair has been horizontally flipped, to correspond to its position in the acquired images. Each image was then reduced in resolution and aligned to correspond, as well as possible, to the acquired images.

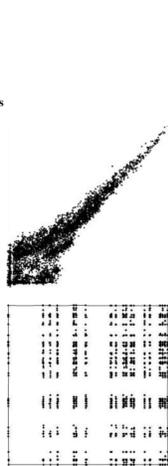
Separation of nonlinear image mixtures

When acquiring an image of a printed document, the image printed on the opposite page often shows through, due to partial transparency of the paper. Here we are dealing with quite a strong case of that effect, because we're using onion skin paper which is quite transparent.

The mixture that is obtained is rather nonlinear, as can be observed from the top figure on the right, which shows a scatter plot of the intensities of corresponding pairs of points from the two pages of a printed document. The scatter plot of the original images, shown in the bottom figure, filled a square, and had only a relatively small number of discrete intensity levels for each image. The fact that the shape of the scatter plot of Fig. 1 is very different from a parallelogram shows that the mixture was strongly nonlinear. The fact that this scatter plot becomes quite narrow in the upper-right corner (which corresponds to the lighter intensities in both images) indicates that, for those intensities, the mixture is close to singular. Finally, the fact that the discrete levels of Fig. 2 became largely blurred in Fig. 1 is due to noise in the process. The process leading from the sources to the observations involved printing the images, on both sides of a sheet of onion skin paper, at 1200 dpi, with a black and white laser printer (with the inherent halftoning of gray levels), and then scanning both sides of the printed sheet at 100 dpi. The noise is due, at least, to the printing process (including the halftoning), to the scanning process and to the non-uniformity in the onion skin paper, especially in its transparency.

The purpose of separation is to recover, from the mixed images that are obtained by scanning both faces of the printed document, the images that had been printed in each of its faces, with as little interference from the other image as possible.

In this example we are creating mixtures that involve natural images, printed text and graphs. The special



1. Introduction

Within the area of unsupervised learning, a problem that has been receiving increasing attention is the one of transforming a set of patterns into new patterns whose components are mutually statistically independent.

Consider that we are given d -dimensional input data vectors $\mathbf{x}=(x_1, x_2, \dots, x_d)$ obeying a probability distribution with density $p_{\mathbf{x}}$. In general, the various components x_i of the data will be statistically interdependent. The problem that we wish to address consists of finding output vectors

$$\mathbf{y}=(y_1, y_2, \dots, y_{d'})=f(\mathbf{x}) \tag{1}$$

such that the output components y_i are mutually independent.

If $d'=d$ and f is invertible, we are simply recoding the data without any loss of information. If $d'<d$ we are reducing the amount of information present in the data. In the latter case, we usually wish to ensure that the extracted features y_i are the most important ones, in some appropriate sense.

In this paper we will discuss the first situation, $d'=d$. If the output components are independent, then

$$p_{\mathbf{y}}(\mathbf{y})=\prod_{i=1}^{d'} p_{y_i}(y_i) \tag{2}$$

i.e., the probability density can be factored into a product of the marginal densities of the output components.

If we assume that the data \mathbf{x} result from a linear combination of independent components, then we can restrict the function f to be linear.

There are several reasons for the growing interest that independent component analysis has been receiving in recent years:

- It can afford a means to perform *source separation*. Assuming that the observed data \mathbf{x} result from an unknown transformation of independent variables \mathbf{z} , i.e.

$$\mathbf{x}=g(\mathbf{z}) \tag{3}$$

where the $\mathbf{z}=(z_1, z_2, \dots, z_k)$ are unknown *source*, one may ask whether the independent output components y_i that we obtain will coincide with the original z_i . We will discuss this issue ahead.



Figure 3: The fourth and fifth pairs of source images, before printing. One image in the last pair has been horizontally flipped. In the fourth pair no flipping has been performed, in order to keep the text's readability. Note, however, that the right-hand image of that pair appears flipped in the mixtures shown ahead.

3. The third pair consists of an image of a natural scene, on one side of the paper, and an image of printed text (Times New Roman, 12-point font) on the other side. Since the text has many large changes of intensity in very small areas, a good “mixing” of the intensities from both images takes place, and the two sources are approximately independent.
4. The fourth pair consists of printed text on both sides of the paper, with a few graphs on one of the sides. Once again, the intensities from the two sides of the paper are well mixed, and therefore approximately independent. The peculiarity of this pair is that, since printed text has a much larger area of white than of black, only a very small percentage of pixels is simultaneously dark on both sides of the paper. This has some influence on the separation results that are obtained, as we shall see.
5. The fifth pair consists of images of natural scenes that have large areas with quasi-uniform intensity. This causes a relatively weak mixing of intensities, making the intensities from the two sides of the paper non-independent. This fact has some impact on the separation results, as we shall see. The large, relatively uniform areas of the images make imperfect separation easier to notice visually than in case 2 above.

The leftmost columns of Figs. 4 and 5 illustrate the joint distributions of the source images. These plots deserve some comments. First of all we should note that, for the joint distributions of the two sources of each pair to be meaningful, the source images had to be adjusted in resolution and aligned, so as to be in the same relative position as in the acquired mixtures. For that purpose each source image was reduced in resolution to the same size as the corresponding acquired mixture images, and was then aligned with the corresponding separated component from nonlinear separation (see Section 4.3 for the alignment procedure and Section 5.2 for the nonlinear separation procedure). Both the resizing and the alignment procedures involved bicubic interpolation of the pixel intensities. The result of such interpolation is visible in the edges of the bars and of the text characters, in Figs. 2 and 3, which show the source images after resizing and alignment.

Some more comments are useful for a better understanding of the source distributions:

- The “grid” look of the first scatter plot reflects the fact that each of the source images had only 25 equally spaced intensities. Some intermediate intensities also appear in the plot due to the intensity interpolation performed in the resizing and alignment processes.
- The second scatter plot shows that, in this case, the two sources are almost independent from each other. The plot shows some evidence of saturation in the lightest intensities of the right-hand source image (vertical axis of the scatter plot). Since this saturation is in the source image, before printing, it should have no significant influence on the mixture and separation processes.
- The third and fourth scatter plots also show that the corresponding source pairs are approximately independent. The distributions of the sources that are images of text show that a very large percentage of their pixels is white. The non-white pixels show a continuous distribution, instead of just a black level, due to the interpolation performed in the resizing and alignment processes. The interpolation effect is much more noticeable here than in the first image pair because, the character sizes being much smaller than the widths of the bars, many pixels fell on black-white edges, and only a very small percentage fell completely within black regions of the characters.

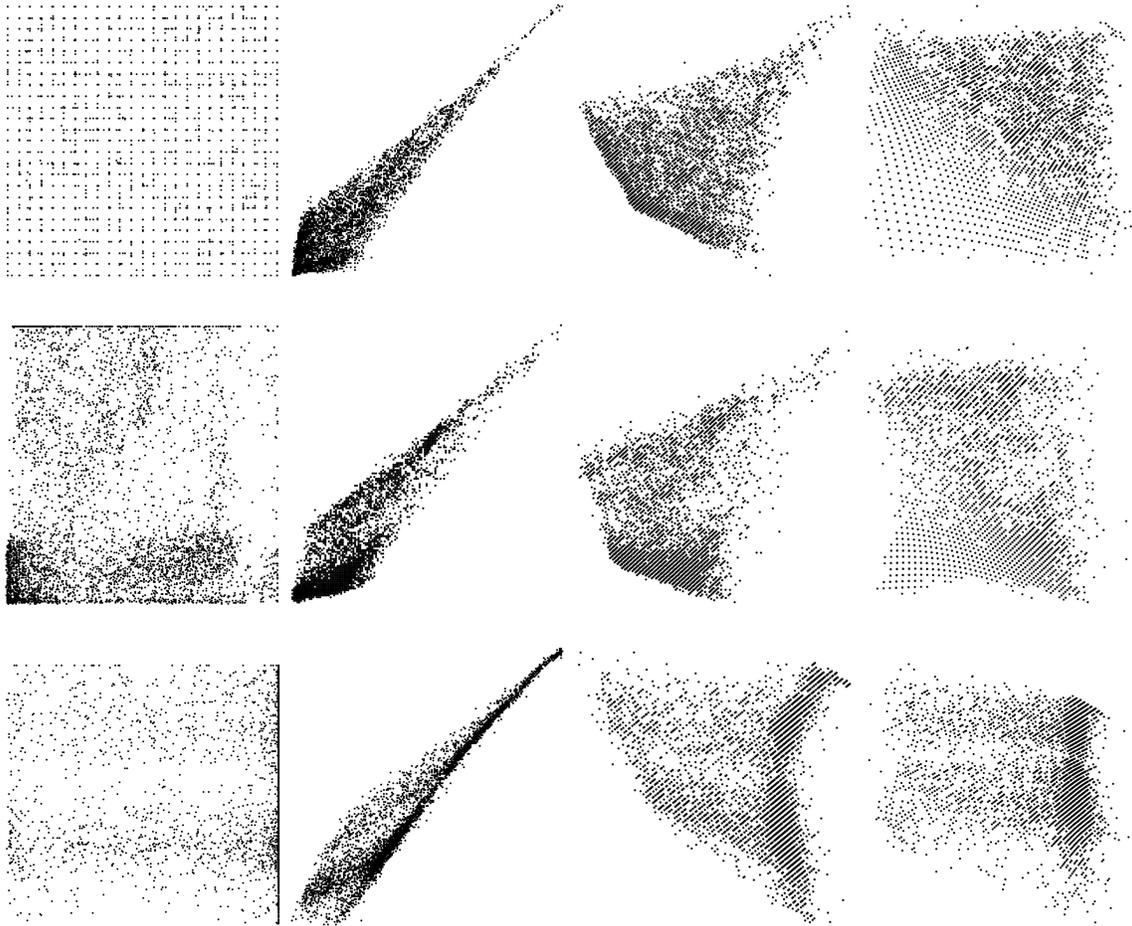


Figure 4: Scatter plots of the first three image pairs. From left to right: source images, acquired images, linear separation and nonlinear separation. The three rows correspond to the three pairs of images of Fig. 2. In each scatter plot, the horizontal axis corresponds to intensities from the left-hand image and the vertical axis to intensities from the right-hand image. The scale of each plot ranges from black (left/bottom) to white (right/top). Each scatter plot shows 5000 randomly selected points from the corresponding pair of images.

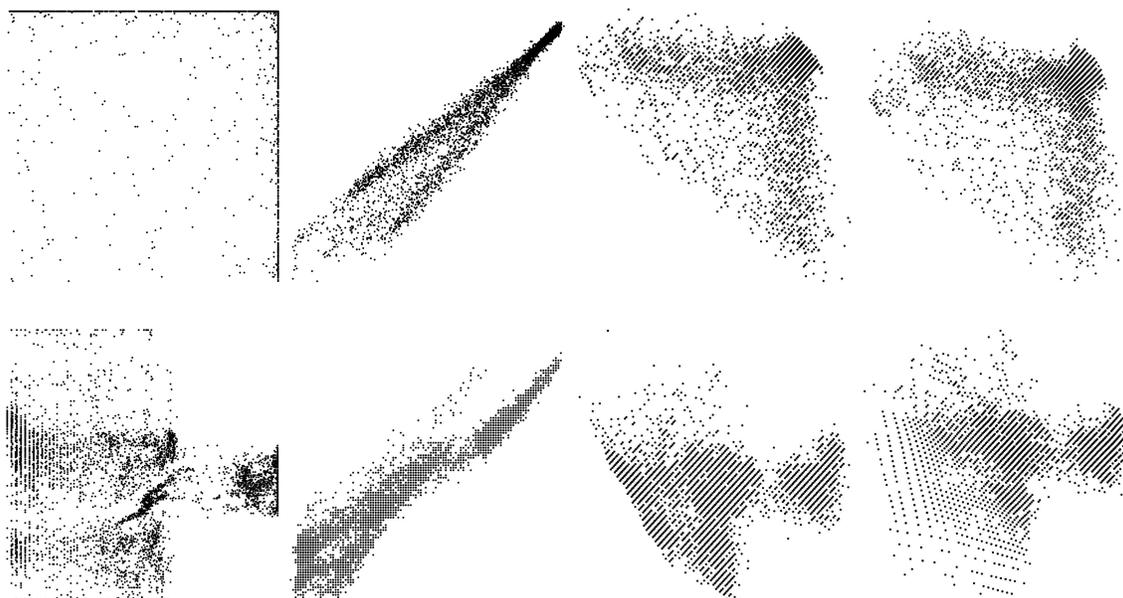


Figure 5: Scatter plots of the fourth and fifth image pairs. From left to right: source images, acquired images, linear separation and nonlinear separation. The two rows correspond to the two pairs of images of Fig. 3.

- The fifth scatter plot clearly shows that the sources of this pair are not independent. The plot shows some evidence of intensity quantization in the darkest levels of the left-hand source image (horizontal axis of the scatter plot), and of saturation in the lightest intensities of the same image. Since the quantization and saturation are in the source image, before printing, they should have no significant influence on the mixture and separation processes.

4.2 The Mixture Process: Printing and Acquisition

The images from each pair were printed on opposite faces of a sheet of onion skin paper. Printing was done with a 1200 dpi laser printer, using the printer’s default halftoning system. Both faces of the sheet of onion skin paper were then scanned with a desktop scanner at a resolution of 100 dpi. This low resolution was chosen on purpose, so that the printer’s halftoning grid would not be apparent in the scanned images. The scanner’s “descreening” option (whose purpose is to minimize the visibility of the halftoning grid) was turned on.

We tried to keep the printing and acquisition processes as symmetrical as possible: the two source images in each pair were handled in an identical way, and the two acquired mixture images in each pair were also handled in an identical way. This implied disabling the scanner’s “automatic image adjustment” feature, which adjusts the acquired image’s brightness, contrast and gamma value in a manner that is not specified in the scanner’s documentation.

The second column of scatter plots of Figs. 4 and 5 shows the joint distributions of the mixture components (after alignment, which is discussed in the next section). The shapes of the mixture distributions show that the mixtures are nonlinear. This is especially clear in the first image pair,

in which the joint distribution of the sources is approximately uniform within a square. A linear mixture process would have resulted in a mixture uniformly distributed within a parallelogram. The observed distribution has a shape that is far from a parallelogram and that is non-uniform, being more dense toward darker intensities than toward lighter ones. Both facts indicate that the mixture is nonlinear. The deviation from a parallelogram shape gives an idea of the amount of nonlinearity.

The mixture distribution, in the first pair, shows no traces of the discrete intensity levels that were present in the source images. This is due to noise introduced by the mixture process. This noise comes from three sources, at least: (1) the printing process, with the halftoning to reproduce grayscale levels; (2) the noise from the scanning process (from other tests of the same scanner this noise appears to be rather weak, essentially amounting to the intensity quantization into 256 levels), and (3) inhomogeneity of the onion skin paper (from our experience this appears to be the strongest source of noise). Later we'll have the possibility to have a better idea of the total amount of noise introduced by the mixture process.

On close inspection, the mixture scatter plots show that the points are arranged on a square grid. This is a result of the intensity quantization performed by the scanner.

4.3 Preprocessing

In the preprocessing stage, in each pair of acquired images one of them was first horizontally flipped, so that both images would have the same orientation. Then the images of each pair were aligned with each other by hand. In preliminary tests we found that even a very careful alignment, using translation, rotation and shear operations on the whole images, could not perform a good simultaneous alignment of all parts of the images. This was probably due to slight geometrical distortions introduced by the scanner. It indicated that an automatic, local alignment was needed. The use of the automatic local alignment relaxed the demands placed on the initial manual alignment.

In the alignment procedure that was finally adopted, the first step consisted just of a manual displacement of one of the images by an integer number of pixels in each direction, so that the two images would be coarsely aligned with each other. In a second step an automatic, local alignment was performed. For this, the resolution of both images was first increased by a factor of 4 in each direction, using bicubic interpolation. Then, one of the images was divided into 100×100 pixel squares (corresponding to 25×25 pixels in the original image), and for each square the best displacement was found, based on the maximum of the cross-correlation with the other image. The whole image was then rebuilt, based on these optimal displacements, and its resolution was reduced by a factor of 4. In this way a local alignment with a resolution of $1/4$ pixel was achieved. Note that, although the alignment consisted only of local translations, it did handle the small rotations and shears that occur in problems of this kind, because these deformations consist just of different displacements for different points of the image. The fact that we used the same displacement for each 25×25 subimage caused only a negligible misalignment, relative to the true displacement that would be appropriate for each pixel.

There is a large variety of image alignment methods described in the literature, varying due to such aspects as the kinds of images to be aligned, the purpose of the alignment, etc. The reader can find an overview, somewhat oriented toward medical images, in (Maintz, 1998). The method that we used was designed specifically for handling the problem we needed to solve, but bears strong resemblances to some of the methods mentioned in that overview, and we make no claims to its originality.

As a final preprocessing step, the intensity range of each pair of images was normalized to the interval $[0, 1]$, 0 corresponding to the darkest pixel in the image pair and 1 to the lightest one. Figures 6 and 7 show the acquired images after preprocessing.²

As said above, we tried to keep the processing of both images in each pair as symmetrical as possible. An obvious asymmetry is due to the fact that only one image in each pair was modified in the alignment procedure. We used a high quality intensity interpolation method (bicubic) in the alignment procedure, so as to affect the image's quality as little as possible. The separation results that we present ahead, based on a symmetry constraint, seem to confirm that the mixture process was kept very close to symmetrical, despite the asymmetry in the alignment procedure.

5. Separation Results

One of the main purposes of the work reported in this paper was to assess the viability and the advantage of performing nonlinear source separation, in a real-life nonlinear mixture problem, by means of an ICA-based separation system. Therefore we used source separation by linear ICA as a baseline for comparison. The next sections present the results of separation by linear and nonlinear ICA, followed by an assessment of the results with objective quality measures.

The mixture process that we used was as symmetrical as possible, so that an exchange of the source images should result just in a corresponding exchange of the mixture images (apart from noise). Therefore we applied symmetry constraints to the separation systems, as detailed ahead.

5.1 Linear Separation

The linear ICA method that we used was MISEP with a linear \mathbf{F} block, which corresponds to INFOMAX with adaptive nonlinearities. Each ψ block was formed by an MLP with a single input and a single output, and with a hidden layer of 20 sigmoidal units. The output unit of each of these MLPs was linear, and there were no "shortcut" connections between input and output. The training set consisted of 5000 pairs of intensities, from randomly chosen pixel pairs of the acquired images. The \mathbf{F} block was initialized with the identity matrix, and training was performed during 200 epochs, which were sufficient for convergence. The \mathbf{F} block was constrained to be symmetrical. Symmetry was not enforced on the ψ blocks because the distributions of the two sources were, in general, different from each other.

For each image pair, ten runs of the separation were made. These differed from one another in the selection of the 5000 pairs of pixels used to form the training set, and in the random initialization of the weights of the ψ MLPs. The results of the ten runs were very similar to one another. Figures 8 and 9 show the results that were best, according to quality measure Q_2 (see Section 5.3). We see that a reasonable degree of separation was achieved in all cases, but some interference remained. The scatter plots in Figs. 4 and 5 (third column) show that, although a certain amount of separation was achieved, the nonlinear character of the mixture could not be undone by linear ICA, as expected. Note: The arrangement of the scatter plots' points into lines (and, in fact, into a grid-like structure, although that is less apparent) is a result of the intensity quantization performed by the scanner.

2. All images of mixtures and of separation results displayed in this paper were adjusted in brightness and contrast so as to saturate the 1% brightest and 1% darkest pixels. This is a procedure that is commonly used for better display of images. This adjustment was performed for image display only: not for image separation and also not for the computation of quality measures.

SEPARATING A REAL-LIFE NONLINEAR IMAGE MIXTURE

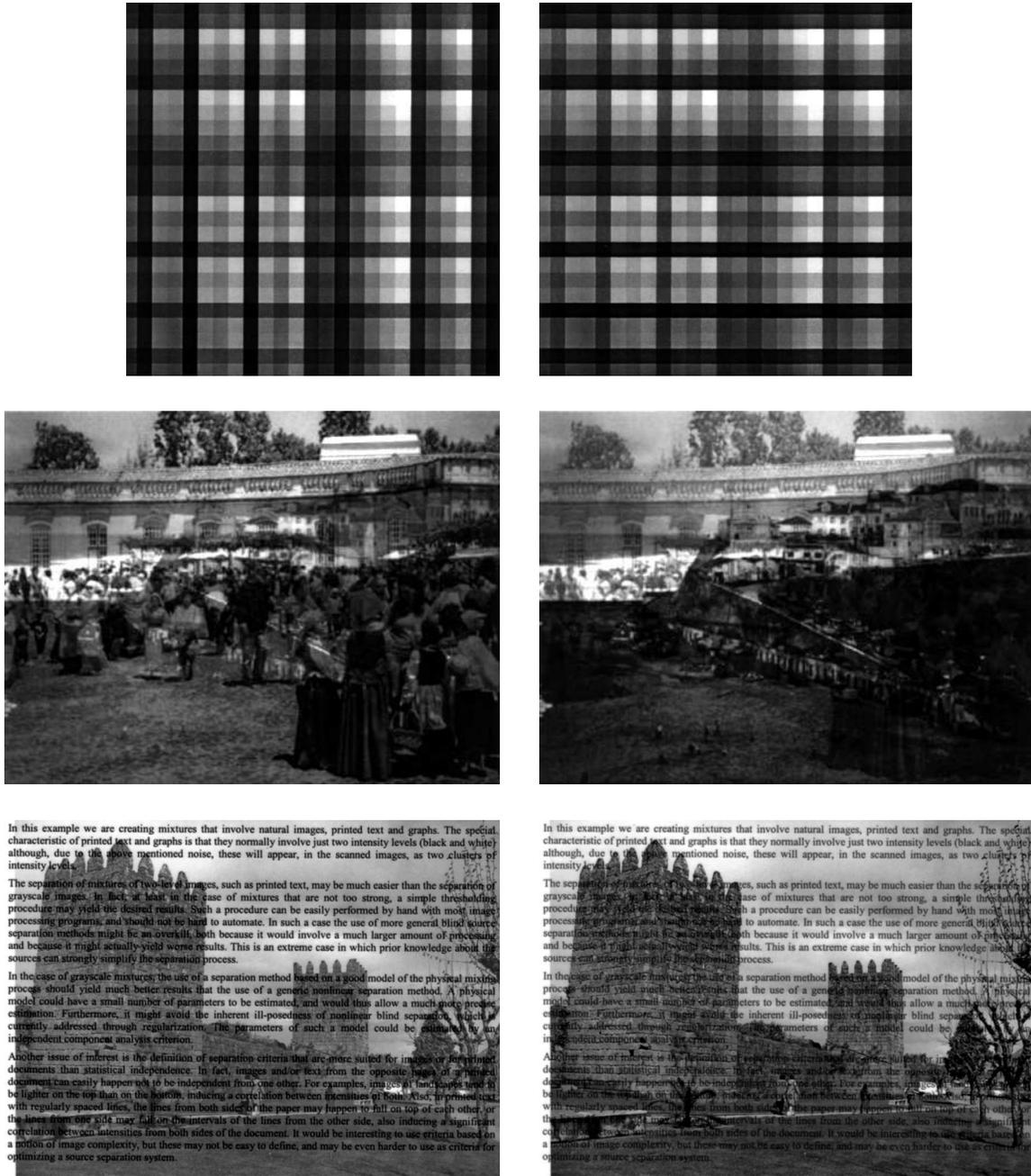


Figure 6: The first three pairs of acquired images, after preprocessing.

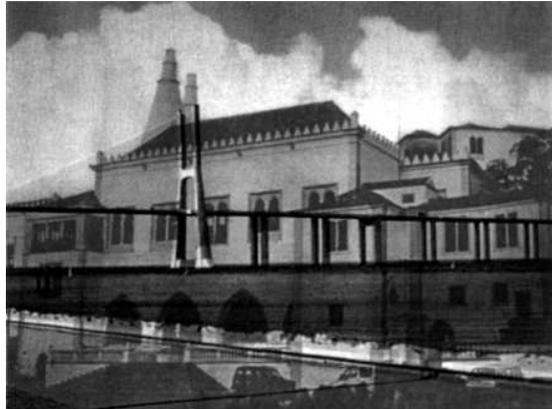
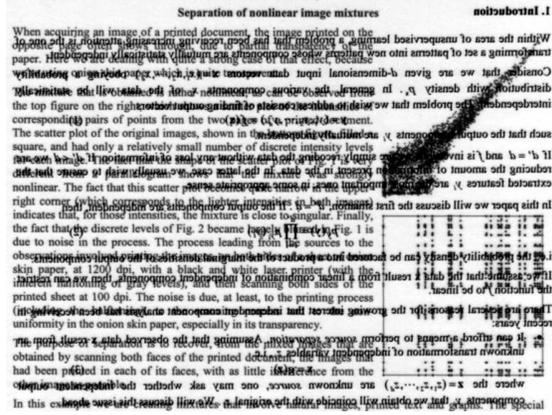
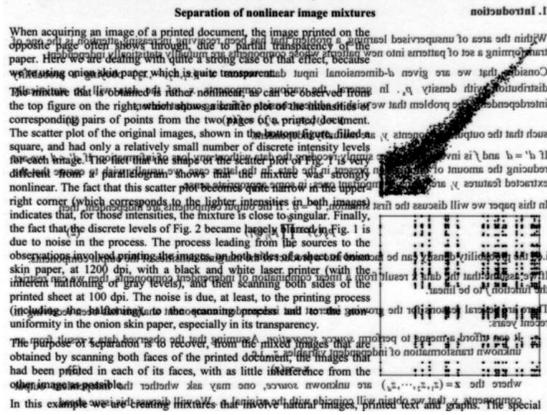
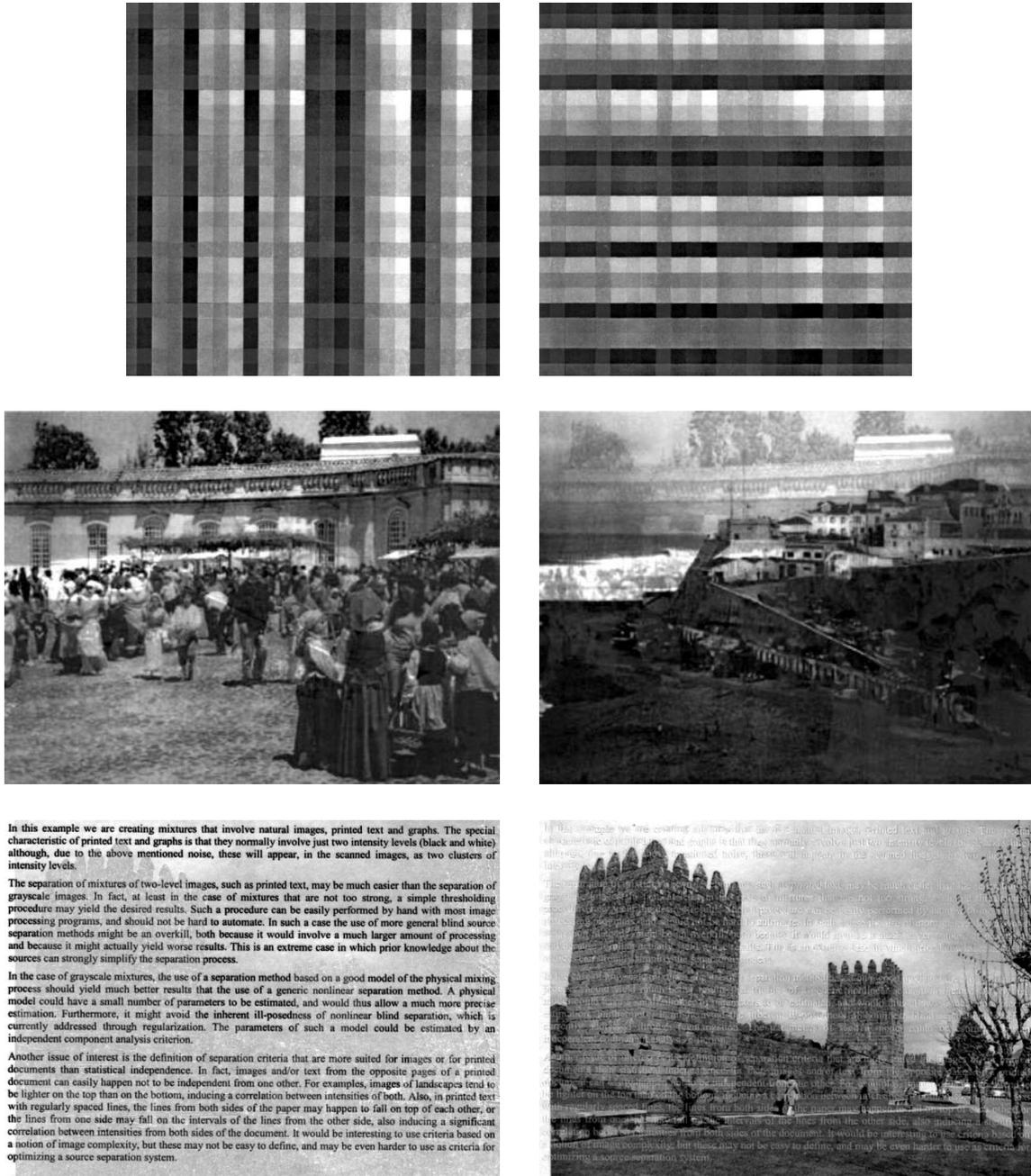


Figure 7: The fourth and fifth pairs of acquired images, after preprocessing.

SEPARATING A REAL-LIFE NONLINEAR IMAGE MIXTURE



In this example we are creating mixtures that involve natural images, printed text and graphs. The special characteristic of printed text and graphs is that they normally involve just two intensity levels (black and white) although, due to the above mentioned noise, these will appear, in the scanned images, as two clusters of intensity levels.

The separation of mixtures of two-level images, such as printed text, may be much easier than the separation of grayscale images. In fact, at least in the case of mixtures that are not too strong, a simple thresholding procedure may yield the desired results. Such a procedure can be easily performed by hand with most image processing programs, and should not be hard to automate. In such a case the use of more general blind source separation methods might be an overkill, both because it would involve a much larger amount of processing and because it might actually yield worse results. This is an extreme case in which prior knowledge about the sources can strongly simplify the separation process.

In the case of grayscale mixtures, the use of a separation method based on a good model of the physical mixing process should yield much better results than the use of a generic nonlinear separation method. A physical model could have a small number of parameters to be estimated, and would thus allow a much more precise estimation. Furthermore, it might avoid the inherent ill-posedness of nonlinear blind separation, which is currently addressed through regularization. The parameters of such a model could be estimated by an independent component analysis criterion.

Another issue of interest is the definition of separation criteria that are more suited for images or for printed documents than statistical independence. In fact, images and/or text from the opposite pages of a printed document can easily happen not to be independent from one another. For examples, images of landscapes tend to be lighter on the top than on the bottom, inducing a correlation between intensities of both. Also, in printed text with regularly spaced lines, the lines from both sides of the paper may happen to fall on top of each other, or the lines from one side may fall on the intervals of the lines from the other side, also inducing a significant correlation between intensities from both sides of the document. It would be interesting to use criteria based on a notion of image complexity, but these may not be easy to define, and may be even harder to use as criteria for optimizing a source separation system.

In this example we are creating mixtures that involve natural images, printed text and graphs. The special characteristic of printed text and graphs is that they normally involve just two intensity levels (black and white) although, due to the above mentioned noise, these will appear, in the scanned images, as two clusters of intensity levels.

The separation of mixtures of two-level images, such as printed text, may be much easier than the separation of grayscale images. In fact, at least in the case of mixtures that are not too strong, a simple thresholding procedure may yield the desired results. Such a procedure can be easily performed by hand with most image processing programs, and should not be hard to automate. In such a case the use of more general blind source separation methods might be an overkill, both because it would involve a much larger amount of processing and because it might actually yield worse results. This is an extreme case in which prior knowledge about the sources can strongly simplify the separation process.

In the case of grayscale mixtures, the use of a separation method based on a good model of the physical mixing process should yield much better results than the use of a generic nonlinear separation method. A physical model could have a small number of parameters to be estimated, and would thus allow a much more precise estimation. Furthermore, it might avoid the inherent ill-posedness of nonlinear blind separation, which is currently addressed through regularization. The parameters of such a model could be estimated by an independent component analysis criterion.

Another issue of interest is the definition of separation criteria that are more suited for images or for printed documents than statistical independence. In fact, images and/or text from the opposite pages of a printed document can easily happen not to be independent from one another. For examples, images of landscapes tend to be lighter on the top than on the bottom, inducing a correlation between intensities of both. Also, in printed text with regularly spaced lines, the lines from both sides of the paper may happen to fall on top of each other, or the lines from one side may fall on the intervals of the lines from the other side, also inducing a significant correlation between intensities from both sides of the document. It would be interesting to use criteria based on a notion of image complexity, but these may not be easy to define, and may be even harder to use as criteria for optimizing a source separation system.

Figure 8: "Best" results of linear separation: first three image pairs.

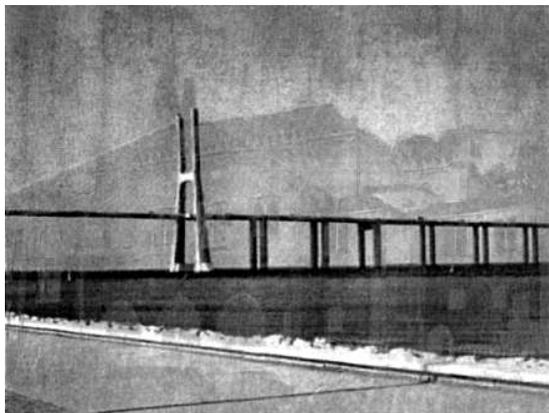
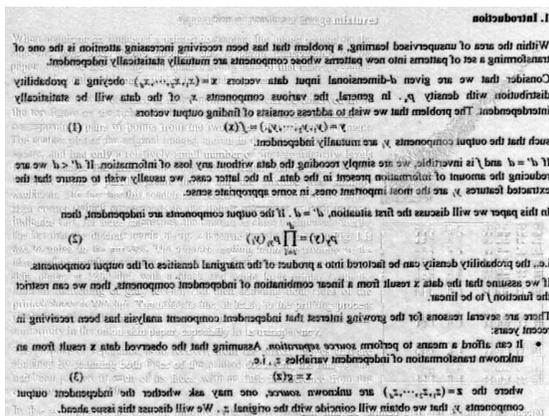
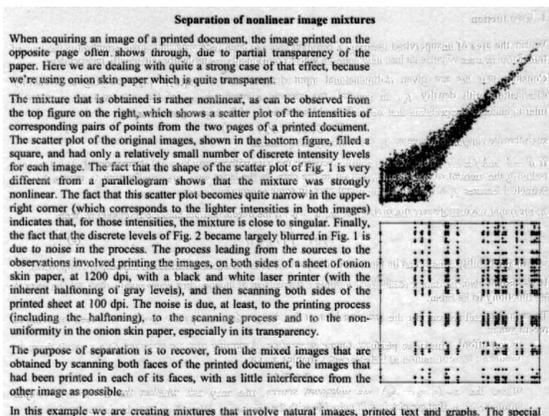


Figure 9: "Best" results of linear separation: fourth and fifth image pairs.

5.2 Nonlinear Separation

For nonlinear separation we used MISEP with a nonlinear \mathbf{F} block. This block consisted of a multilayer perceptron with two inputs, two outputs and a hidden layer of 40 sigmoidal units. The output units were linear, and the hidden units were divided into two groups of 20, each group being connected to one of the output units. This MLP also had direct, “shortcut” connections between inputs and outputs. Since the output units were linear, the block could implement linear separation exactly, by setting the weights of the hidden layer’s connections to zero.

As noted above, regularization plays an important role in dealing with the ill-posedness of nonlinear ICA. In our case regularization was achieved by three means: (i) initializing the \mathbf{F} network to perform an identity mapping, (ii) constraining that network to be symmetrical, and (iii) constraining that network to be linear during the first 100 training epochs (by keeping the output weights of the hidden layer equal to zero during those epochs). Training was stopped at 400 epochs. At that point the progress of the optimization was in general very slow. As a test, in a few cases the optimization was extended to a much larger number of epochs, without any significant change in the separation results. Therefore the exact stopping point that was chosen doesn’t appear to have had any significant influence on the results. The ψ blocks had the same structure as in the linear separation case. Each 400-epoch training run took approximately 9 minutes on a 1.6 GHz Pentium-M (Centrino) processor.

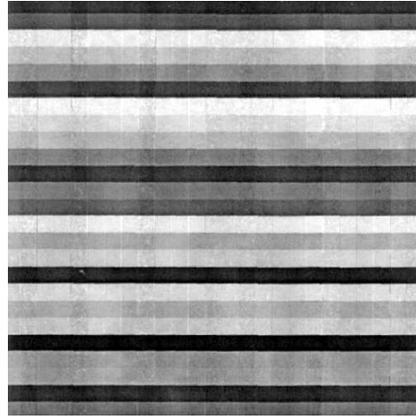
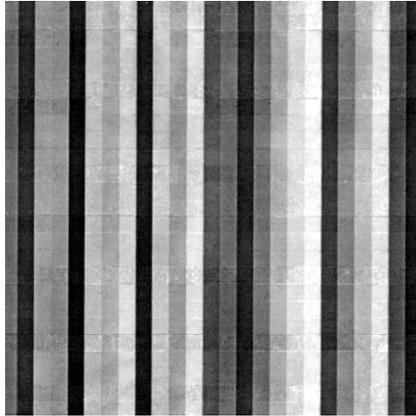
For each image pair, ten runs of the separation were made, with different random selections of the 5000 pixel pairs forming the training set, and with different random initializations of the MLPs’ weights (excluding, of course, those weights that were initially set to the identity matrix or to zero). Figures 10 and 11 show the best results that were obtained (“best” according to quality measure Q_2). The scatter plots corresponding to these separations are shown in the rightmost column of Figs. 4 and 5. Figures 12 and 13 show the worst separation results that were obtained (“worst” again according to Q_2).

5.3 Measures of Separation Quality

The images shown in the previous section give an idea of the separation quality, but their evaluation is rather subjective. It depends on the viewer, as well as on other factors such as the conditions under which the images were printed or are viewed. Furthermore, a reasonable amount of image superposition can pass unnoticed in regions in which the “main” image has much variability. For these reasons we decided to also use objective measures of separation quality, which are not sensitive to such effects.

Experience with objective quality measures for nonlinear source separation is still very limited. This led us to compute four different quality measures. The first, that we denote by Q_1 , was simply the signal to noise ratio (SNR) of the extracted component relative to the corresponding source.³ We should note that, in a nonlinear separation context, the SNR, besides being sensitive to incomplete source separation and to noise, is also sensitive to any nonlinear transformation of the intensity scale that may be caused by the mixture and separation processes. It is well known that, in linear separation, the sources are recovered with unknown scale factors. In nonlinear ICA-based separation, each recovered source may be subject to an unknown nonlinear, invertible transformation. Measure Q_1 gives a global indication of the distortion of the extracted component relative to the corresponding

3. For the computation of all quality measures we used the resized and aligned source images.



In this example we are creating mixtures that involve natural images, printed text and graphs. The special characteristic of printed text and graphs is that they normally involve just two intensity levels (black and white) although, due to the above mentioned noise, these will appear, in the scanned images, as two clusters of intensity levels.

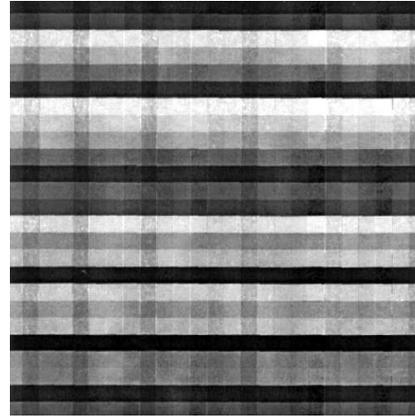
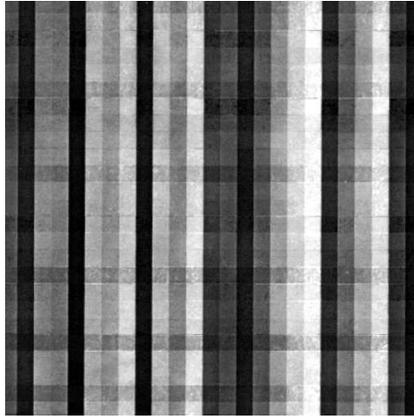
The separation of mixtures of two-level images, such as printed text, may be much easier than the separation of grayscale images. In fact, at least in the case of mixtures that are not too strong, a simple thresholding procedure may yield the desired results. Such a procedure can be easily performed by hand with most image processing programs, and should not be hard to automate. In such a case the use of more general blind source separation methods might be an overkill, both because it would involve a much larger amount of processing and because it might actually yield worse results. This is an extreme case in which prior knowledge about the sources can strongly simplify the separation process.

In the case of grayscale mixtures, the use of a separation method based on a good model of the physical mixing process should yield much better results than the use of a generic nonlinear separation method. A physical model could have a small number of parameters to be estimated, and would thus allow a much more precise estimation. Furthermore, it might avoid the inherent ill-posedness of nonlinear blind separation, which is currently addressed through regularization. The parameters of such a model could be estimated by an independent component analysis criterion.

Another issue of interest is the definition of separation criteria that are more suited for images or for printed documents than statistical independence. In fact, images and/or text from the opposite pages of a printed document can easily happen not to be independent from one other. For examples, images of landscapes tend to be lighter on the top than on the bottom, inducing a correlation between intensities of both. Also, in printed text with regularly spaced lines, the lines from both sides of the paper may happen to fall on top of each other, or the lines from one side may fall on the intervals of the lines from the other side, also inducing a significant correlation between intensities from both sides of the document. It would be interesting to use criteria based on a notion of image complexity, but these may not be easy to define, and may be even harder to use as criteria for optimizing a source separation system.



Figure 10: “Best” results of nonlinear separation: first three image pairs.



In this example we are creating mixtures that involve natural images, printed text and graphs. The special characteristic of printed text and graphs is that they normally involve just two intensity levels (black and white) although, due to the above mentioned noise, these will appear, in the scanned images, as two clusters of intensity levels.

The separation of mixtures of two-level images, such as printed text, may be much easier than the separation of grayscale images. In fact, at least in the case of mixtures that are not too strong, a simple thresholding procedure may yield the desired results. Such a procedure can be easily performed by hand with most image processing programs, and should not be hard to automate. In such a case the use of more general blind source separation methods might be an overkill, both because it would involve a much larger amount of processing and because it might actually yield worse results. This is an extreme case in which prior knowledge about the sources can strongly simplify the separation process.

In the case of grayscale mixtures, the use of a separation method based on a good model of the physical mixing process should yield much better results than the use of a generic nonlinear separation method. A physical model could have a small number of parameters to be estimated, and would thus allow a much more precise estimation. Furthermore, it might avoid the inherent ill-posedness of nonlinear blind separation, which is currently addressed through regularization. The parameters of such a model could be estimated by an independent component analysis criterion.

Another issue of interest is the definition of separation criteria that are more suited for images or for printed documents than statistical independence. In fact, images and/or text from the opposite pages of a printed document can easily happen not to be independent from one other. For examples, images of landscapes tend to be lighter on the top than on the bottom, inducing a correlation between intensities of both. Also, in printed text with regularly spaced lines, the lines from both sides of the paper may happen to fall on top of each other, or the lines from one side may fall on the intervals of the lines from the other side, also inducing a significant correlation between intensities from both sides of the document. It would be interesting to use criteria based on a notion of image complexity, but these may not be easy to define, and may be even harder to use as criteria for optimizing a source separation system.

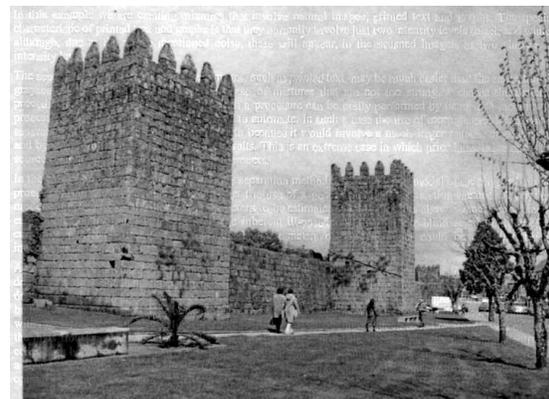


Figure 12: “Worst” results of nonlinear separation: first three image pairs.

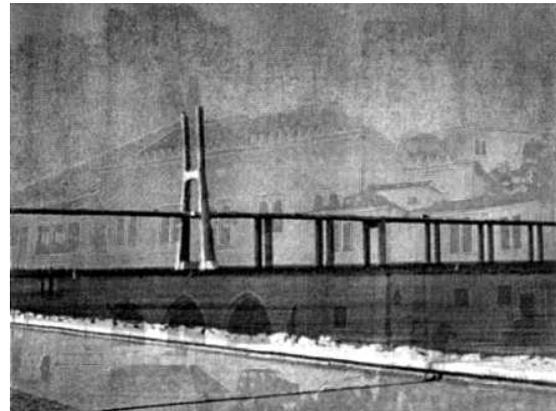
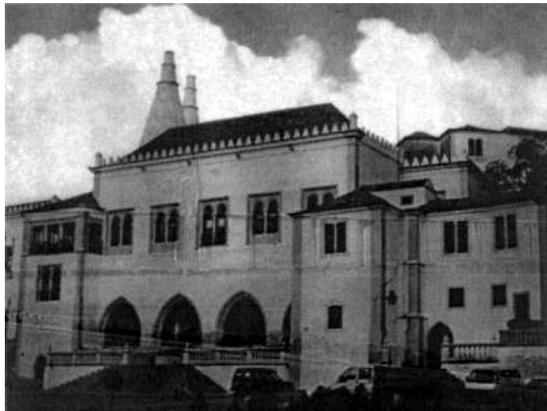
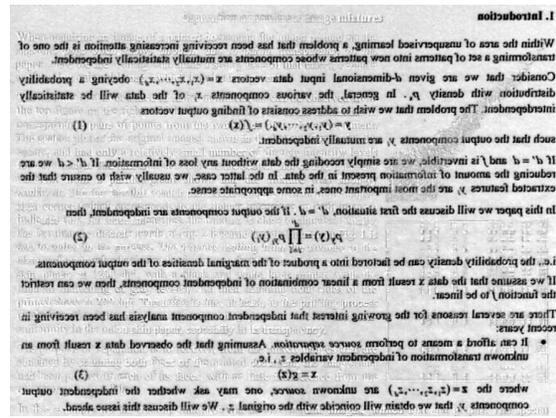
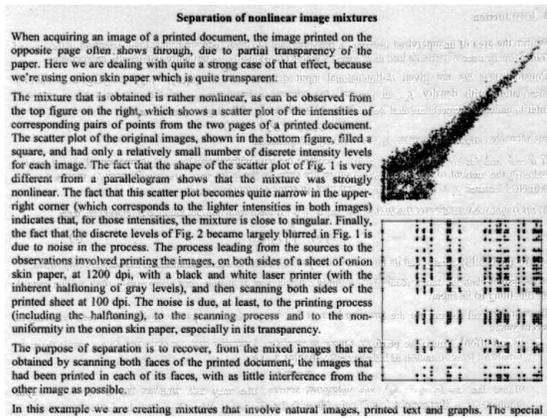


Figure 13: “Worst” results of nonlinear separation: fourth and fifth image pairs.

source, including any nonlinear transformation of the intensity scale, besides including incomplete separation and noise.

Due to the possible presence of a nonlinear transformation of the intensity scale, our other three quality measures were defined so as to be invariant to such transformations. The second quality measure, Q_2 , was a signal to noise ratio, modified so that it had the invariance property mentioned above. It was given by

$$Q_2 = \frac{\text{variance of } S}{\text{variance of } N}, \quad (4)$$

where S was the source image and N was the noise that was present in the extracted component. This noise was computed as

$$N = f(Y) - S, \quad (5)$$

Y being the extracted component, and f being a nonlinear, monotonic transformation chosen so that Q_2 was maximal. In other terms, we chose a nonlinear, monotonic transformation of the intensity scale of the extracted component that made it become as close as possible to the corresponding source in SNR terms, and then used its SNR as the quality measure. The optimal $f(\cdot)$ was computed in table form. This was possible because the number of intensity levels in each image is finite, since each image has a finite number of pixels.

The other two measures that we used were information-theoretic:

- Q_3 was the mutual information between each extracted component and the corresponding source. The mutual information was estimated from a set of 5000 randomly selected pixel pairs, chosen independently from those forming the training set, and was computed using the $I^{(1)}$ estimator described in (Kraskov et al., 2004), with $k = 3$ (k is the nearest neighbor order used in that estimation algorithm; its recommended range, given in that reference, is between 2 and 4).
- Q_4 was the mutual information between each extracted component and the opposite source, computed in the same manner as for Q_3 .

Note that other quality measures could easily be envisaged. For example, $Q_4 - Q_3$ would be a measure similar in spirit to the well known Amari index (Amari et al., 1996), but based on mutual information, to account for nonlinearities, and using a difference instead of a quotient due to its logarithmic character.

Another kind of measure that might come to mind would be similar to Q_4 (indicating the amount of interference, from the “wrong” source, that is present in the extracted component) but measured in terms of SNR instead of mutual information. Such a measure would not have made much sense, however, because in a nonlinear context the interference can be “positive” in some parts of the image and “negative” in other parts. These positive and negative parts would tend to cancel out. Therefore such a measure could sometimes indicate a misleadingly low amount of interference. In a measure like Q_4 , based on mutual information, such positive and negative interferences do not cancel out, but instead have a cumulative effect.

As a reference for assessing the amount of separation achieved by the various methods, we show in Table 1 the values of the quality measures for the mixture components after preprocessing, without any separation.

The mean values of the quality measures for each of the ten-run series of separations are shown in Table 2. Note that for Q_1 , Q_2 and Q_3 higher values are best, while for Q_4 lower values are best.

Image pair	Quality measure	No separation	
		source 1	source 2
1	Q_1	1.9	1.9
	Q_2	6.2	6.2
	Q_3	1.21	1.23
	Q_4	0.48	0.49
2	Q_1	-1.7	6.0
	Q_2	3.7	8.9
	Q_3	1.11	1.34
	Q_4	0.56	0.60
3	Q_1	-4.5	6.6
	Q_2	3.8	8.1
	Q_3	0.38	1.65
	Q_4	1.35	0.12
4	Q_1	0.9	-2.3
	Q_2	5.6	3.3
	Q_3	0.56	0.29
	Q_4	0.23	0.43
5	Q_1	9.6	-6.4
	Q_2	11.7	2.7
	Q_3	1.85	1.07
	Q_4	0.86	1.18

Table 1: Values of the objective quality measures for the unseparated mixture components. In this and in the following table Q_1 and Q_2 are given in dB and Q_3 and Q_4 in bits.

Image pair	Quality measure	Linear separation		Nonlinear separation	
		source 1	source 2	source 1	source 2
1	Q_1	9.0	8.7	13.8	13.1
	Q_2	11.9	11.6	14.7	14.2
	Q_3	2.03	1.96	2.45	2.39
	Q_4	0.48	0.46	0.23	0.26
2	Q_1	5.2	10.5	9.3	13.9
	Q_2	8.1	12.9	11.0	15.0
	Q_3	1.56	1.78	1.83	1.95
	Q_4	0.37	0.53	0.24	0.40
3	Q_1	4.5	11.2	6.2	11.2
	Q_2	7.8	12.4	9.1	13.8
	Q_3	0.80	1.99	0.85	2.11
	Q_4	0.36	0.18	0.09	0.15
4	Q_1	5.8	3.4	6.0	3.7
	Q_2	8.8	6.7	9.1	7.1
	Q_3	0.74	0.48	0.75	0.51
	Q_4	0.11	0.16	0.11	0.16
5	Q_1	13.4	6.6	14.2	6.4
	Q_2	14.7	7.9	15.3	7.8
	Q_3	2.13	1.34	2.19	1.29
	Q_4	0.71	0.46	0.56	0.49

Table 2: Objective quality results. The results shown are the average for each of the sets of ten test runs. For each pair (linear and nonlinear, for the same source), the best result is shown in bold if the difference was significant at the 95% confidence level. For Q_1 , Q_2 and Q_3 higher results are better, while for Q_4 lower results are better.

The cases in which the difference between linear and nonlinear separation was significant at the 95% confidence level are shown in bold in the table.

The measure that seemed to correlate best with our subjective evaluation of separation quality was Q_2 , and this is why we chose it for the selection of the “best” and “worst” examples shown in Sections 5.1 and 5.2. The next best was Q_1 . Q_4 , which was intended to measure the amount of interference from the “wrong” source, was the one which correlated worst with our subjective quality evaluation.

5.4 Assessment of the Results

For the first three image pairs, both the objective quality measures and our subjective evaluation showed a clear advantage of nonlinear separation over linear separation. Even the worst results of nonlinear separation seemed to be better, in general, than the best results of linear separation. Comparison of the third and fourth columns of scatter plots (Figs. 4 and 5) also confirms the advantage of nonlinear separation. This advantage was not so clear, however, for the fourth and fifth image pairs. We discuss now why we think this was so.

For the fourth image pair, most objective quality measures still show an advantage of nonlinear separation, but this advantage is very small, and our subjective evaluation showed the results of linear and nonlinear separation to be very similar in quality. This is also confirmed by comparing the corresponding scatter plots in Figs. 4 and 5. In this image pair, most pixels are white in at least one of the sources. The source scatter plot is dominated by two lines of points, located on the top and right-hand edges of the plot. This has the consequence that, with the specific mixture that was involved in the problem under study, linear ICA was able to perform a rather good separation. We see from the scatter plot of the linearly separated components that the lower-left area, corresponding to simultaneously dark pixels on both sources, was left unfilled by linear ICA. But this represented a rather small percentage of pixels, and had little impact on the overall separation quality.

We also see, from the rightmost scatter plot, that nonlinear separation also left the lower-left area unfilled. This may seem to be due to an incomplete optimization, but we tried extending the optimization to a much larger number of epochs without any significant change in the results. It is possible that the result shown corresponds to a local optimum. By playing with the network structure, with the initial conditions and with the constraints, we were sometimes able to get a result in which the lower left area of the scatter plot was filled. However, this made very little difference in the subjective or objective quality of the separation.

The results for the fifth image pair show that one of the sources was best separated by the linear method, while the other was best separated by the nonlinear one. But the differences between the two methods were rather small, even though most of them were statistically significant. Nonlinear separation apparently suffered a negative impact from the fact that the sources were not independent from each other and we were using independence as the separation criterion. The nonlinear separation network had many more degrees of freedom than the linear one, and used them to try to make the extracted components more independent from each other. In doing so it impaired the separation of one of the sources, instead of improving it, since the actual sources were not independent.

An important aspect of the results that we obtained is that, although the mixture process was nonlinear, and nonlinear separation could, in principle, introduce an arbitrary nonlinear transformation in each separated component, the total amount of nonlinearity introduced by the mixture and separation processes was relatively small. This is clear from the separation images that were shown (which were only normalized in brightness and contrast, as mentioned above) and from the values of the Q_1 measure. We also illustrate this, in a more clear form, in Fig. 14. This figure shows a scatter plot of the first extracted component versus the corresponding source, for the “average” case of the first image pair (the “average” case was chosen as the one whose value of Q_2 was closest to the average for the ten runs).

From our experience, there were two factors that were important in achieving this low level of nonlinearity. One was the fact that we linearly “primed” the separation network, by constraining it to be linear during the first 100 epochs. The other factor was that we gave a great amount of flexibility to the ψ networks, by implementing them with a large number of hidden units. In previous tests in which these networks had only 6 hidden units, the separation results, as measured by Q_2 , Q_3 or Q_4 were not very different from those presented here, but there often was a significant amount of nonlinearity introduced in the extracted components. This seems to have been caused by the \mathbf{F} block trying to compensate for the limitations of the ψ networks which could not, by themselves, make the distribution of each Z_i close to uniform.

There are some other aspects of the results, and of the experience that we gained in studying this problem, that are worth discussing. One of them has to do with the amount of noise introduced

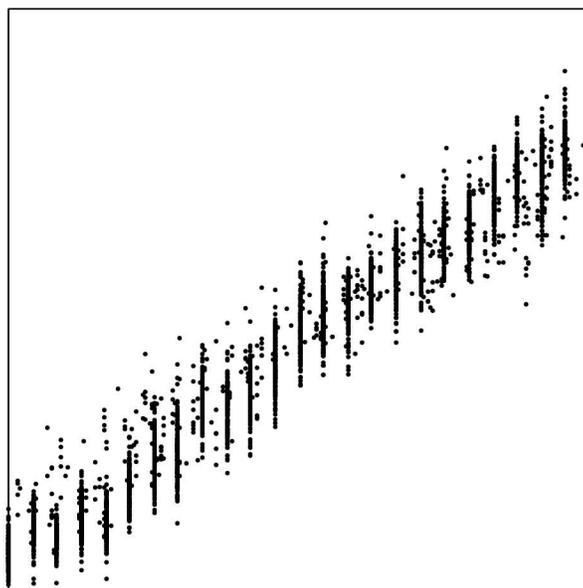


Figure 14: Scatter plot of the first extracted component versus the corresponding source, in an “average” run of nonlinear separation of the first image pair. Horizontal axis: source; vertical axis: extracted component.

by the mixture process. We can take advantage of the fact that the source images that contain text have a large percentage of purely white pixels, which show up as strong, very thin lines in the corresponding scatter plots in the first column of Figs. 4 and 5, for having an idea of the amount of noise present in the mixtures and in the separated components. After the mixture, and also after linear or nonlinear separation, these lines appear broadened in the scatter plots, looking like fuzzy dark bands. The widths of these bands give an idea of the amount of noise that was introduced by the mixing, or by the mixing plus separation. In the separation results the noise represents a significant percentage of the whole intensity range. Note that the separation process does not, by itself, introduce any noise. However, since it essentially consists of performing a weighted difference between the two mixture components, it does increase the amount of noise that is present, in relative terms.

Another interesting aspect has to do with understanding the “scale” of the quality measures based on mutual information (especially of Q_3 since, as we’ve already said, Q_4 seemed to be less meaningful). We were surprised by the relatively low values of mutual information between source and extracted component, even when the images looked well separated and Q_2 indicated relatively high SNR values after compensation of nonlinearities. For natural scene images, the mutual information between source and extracted component was roughly around 2 bits, while for text images it was below 1 bit. We can also observe from Table 2 that, for each source image, a change of 1 dB in SNR (i.e. in Q_2) corresponded, approximately, to a change of 0.1 bit in Q_3 . Small changes in the value of mutual information seem to be much more significant than we expected before performing these tests.

An important aspect of the mixture process, that we have not mentioned so far, is that it didn't seem to be a purely point-wise process. The intensity of each source image at each point appeared to affect the observed mixture intensities in a small neighborhood of that point. This is especially noticeable by closely examining the separation results in the cases in which the image to be suppressed was a text image. The cause of this phenomenon probably was some lateral diffusion of light inside the paper. The effect was relatively weak at the scanning resolution that we used, but should become more pronounced at higher resolutions. A more perfect separation system should take this into account. However, non-point-wise nonlinear ICA is still essentially an unstudied topic, and is beyond the scope of this paper.

Another important aspect has to do with the use of the symmetry constraint. We were careful in ensuring that, both during scanning and in the preprocessing stage, both sides of the paper were handled in the same way. This allowed us to use a symmetry constraint in the separation networks. Such symmetry conditions in the mixture can probably be obtained when using a system like our desktop scanner, in which the paper has to be flipped, and the same set of sensors is used to acquire both sides. However, industrial scanners, which are used to digitize large quantities of documents, normally acquire both sides of the document at the same time, using two different sets of sensors. Such scanners often are strongly non-symmetric. In such cases the symmetry constraint couldn't probably be used, or would have to be used only in an initial part of the training, after which it would have to be relaxed. We had no access to images from such scanners, and therefore couldn't assess what degree of separation would be achievable with them.

Still regarding a possible application to an actual scanning or photocopying device, there are two other aspects worth mentioning. One is that it doesn't seem to be possible to have a fixed separator, optimized at the factory for a specific device. This is because the mixture depends at least on the paper being used, and possibly also on the printing ink, halftoning process and other similar factors. It seems possible, however, to develop a physical model of the mixture process, with a small number of parameters, and then to find (algebraically or by approximate means) a parameterized inverse system. Its parameters may then be estimated through an ICA criterion. MISEP seems suited for this task, since it can use essentially any parameterized nonlinear system in the \mathbf{F} block.

Another practical aspect has to do with the possible warping (existence of ripples) in the document being processed. We found that even very weak ripples, barely noticeable in the scanned images, would result in very strong light and dark bands in the separated images, both with linear and with nonlinear separation. This was, of course, a situation in which the mixture was spatially variant, and could not be adequately undone by a spatially invariant system. In our case we solved the problem by applying a very strong pressure to the cover of the scanner while scanning the documents, in order to eliminate the ripples. This might become an important issue in a practical application.

6. Conclusion

We showed an application of ICA to nonlinear source separation in a real-life problem of practical interest. One of the main issues that have been discussed in the last few years, concerning nonlinear ICA, is whether its inherent ill-posedness can be handled in practical situations. Our results show that it can, at least in this specific problem. We should say, however, that it took quite a bit of experimentation to find a set of conditions that could be used for all image pairs, yielding a good separation with relatively little variability in the separation results. In an earlier work (Almeida

and Faria, 2004) we had not yet been able to achieve an adequate form of regularization, without resorting to an \mathbf{F} block with a specialized form.

We presented comparisons of MISEP-based nonlinear ICA with linear ICA, one of the main purposes being to demonstrate the feasibility and the advantage of nonlinear source separation through ICA in a practical situation. It would also be very interesting to compare the nonlinear separation results presented here with those obtained with other nonlinear separation methods, such as ensemble learning (Lappalainen and Honkela, 2000), kernel-based nonlinear ICA (Harmeling et al., 2003) or geometric ICA (Theis et al., 2003). That comparison would have been outside the scope of the present paper. First of all, it would have involved a very large amount of additional work. Furthermore, the results obtained with a specific method are often much better if the method is tuned by someone experienced in its use. We have a reasonable amount of experience in using MISEP, but virtually no experience with any of the other methods. To enable comparisons we chose to make our test data, as well as our separation routines, available online (see the end of Section 4.3).

Future work will address several different issues, among which we can mention:

- The development of separation criteria that are more adequate for this problem than statistical independence. We have seen that, in this problem, the images to be separated may happen not to be independent. In such a case the quality of separation suffers. A more adequate separation criterion would not cause such degradation and might also be able to overcome much of the ill-posedness of nonlinear ICA, decreasing the dependence on regularization.
- The use of the spatial redundancy of images to reduce the ill-posedness of the problem, hopefully achieving separation with less dependence on regularization. Some published results (Harmeling et al., 2003) suggest that the use of signal structure may help to separate nonlinear mixtures with much reduced ill-posedness. That may make kernel-based nonlinear ICA a good candidate for handling this problem.
- The study of models of the mixture process that involve relatively few parameters. It seems possible to develop physically based and/or empirical models that depend on a few parameters (such as paper transparency and reflectivity, among others). Having few parameters, such models may have no ill-posedness, and may also be able to easily handle non-symmetrical systems.

Acknowledgments

The author would like to acknowledge Alexander Shustorovitch for useful information regarding industrial scanners, and would also like to acknowledge the very useful comments made by the anonymous reviewers.

References

- L. B. Almeida. Faster training in nonlinear ICA using MISEP. In *Proceedings of the International Workshop on Independent Component Analysis and Blind Signal Separation*, pages 113–118, Nara, Japan, 2003a. URL <http://www.lx.it.pt/~lbalmeida/papers/AlmeidaICA03.pdf>.
- L. B. Almeida. MISEP – Linear and nonlinear ICA based on mutual information. *Journal of Machine Learning Research*, 4:1297–1318, 2003b. URL <http://www.jmlr.org/papers/volume4/almeida03a/almeida03a.pdf>.
- L. B. Almeida and M. Faria. Separating a real-life nonlinear mixture of images. In Carlos G. Puntonet and Alberto Prieto, editors, *Independent Component Analysis and Blind Signal Separation (Proc. ICA'2004)*, number 3195 in Lecture Notes in Artificial Intelligence, pages 729–736, Granada, Spain, 2004. Springer-Verlag. URL <http://www.lx.it.pt/~lbalmeida/papers/AlmeidaICA04.pdf>.
- S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In David Touretzky, Michael Mozer, and Mark Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 757–763. MIT Press, 1996. URL <http://www.islab.brain.riken.go.jp/~amari/pub/acyNIPS95.ps.Z>.
- A. Bell and T. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995. URL [ftp://ftp.cnl.salk.edu/pub/tony/bell.blind.ps](http://ftp.cnl.salk.edu/pub/tony/bell.blind.ps).
- G. Burel. Blind separation of sources: A nonlinear neural algorithm. *Neural Networks*, 5(6):937–947, 1992.
- P. Comon. Independent component analysis – a new concept? *Signal Processing*, 36:287–314, 1994.
- G. Darmois. Analyse générale des liaisons stochastiques. *Revue de l'Institut International de Statistique*, 21:2–8, 1953.
- G. Deco and W. Brauer. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8:525–535, 1995.
- M. Haritopoulos, H. Yin, and N. Allinson. Image denoising using SOM-based nonlinear independent component analysis. *Neural Networks*, 15(8–9):1085–1098, 2002.
- S. Harmeling, A. Ziehe, M. Kawanabed, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15:1089–1124, 2003. URL http://ida.first.fraunhofer.de/~harmeli/papers/article_on_ktdsep.pdf.
- A. Hyvärinen and E. Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000. URL <http://www.cs.helsinki.fi/u/ahyvarin/papers/NN00new.pdf>.

- A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999. URL <http://www.cis.hut.fi/~aapo/ps/NN99.ps>.
- C. Jutten and J. Karhunen. Advances in blind source separation (BSS) and independent component analysis (ICA) for nonlinear mixtures. *International Journal of Neural Systems*, 14(5):267–292, 2004. URL <http://www.worldscinet.com/128/14/preserved-docs/1405/S012906570400208X.pdf>.
- A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Physical Review E*, 69:066138, 2004. URL <http://arxiv.org/pdf/cond-mat/0305641>.
- H. Lappalainen and A. Honkela. Bayesian nonlinear independent component analysis by multi-layer perceptrons. In M. Girolami, editor, *Advances in Independent Component Analysis*, pages 93–121. Springer-Verlag, 2000. URL <http://www.cis.hut.fi/harri/ch7.ps.gz>.
- S.-I. Lee and S. Batzoglou. Application of independent component analysis to microarrays. *Genome Biology*, 4(11):R76, 2003. URL <http://genomebiology.com/2003/4/11/R76>.
- J. B. Maintz. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998. URL <http://www.cs.uu.nl/people/twan/personal/media97.pdf>.
- G. C. Marques and L. B. Almeida. Separation of nonlinear mixtures using pattern repulsion. In J. F. Cardoso, C. Jutten, and P. Loubaton, editors, *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation*, pages 277–282, Aussois, France, 1999. URL <http://www.lx.it.pt/~lbalmeida/papers/MarquesAlmeidaICA99.ps.zip>.
- F. Palmieri, D. Mattered, and A. Budillon. Multi-layer independent component analysis (MLICA). In J. F. Cardoso, C. Jutten, and P. Loubaton, editors, *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation*, pages 93–97, Aussois, France, 1999.
- B. Pearlmutter and L. Parra. A context-sensitive generalization of independent component analysis. In *International Conference on Neural Information Processing*, Hong Kong, 1996. URL <http://newton.bme.columbia.edu/~lparra/publish/iconip96.pdf>.
- J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.
- A. Taleb and C. Jutten. Source separation in post-nonlinear mixtures. *IEEE Transactions on Signal Processing*, 47:2807–2820, 1999.
- F. J. Theis, C. G. Puntonet, and E. W. Lang. Nonlinear geometric ICA. In *Proceedings of the International Workshop on Independent Component Analysis and Blind Signal Separation*, pages 275–280, Nara, Japan, 2003. URL http://homepages.uni-regensburg.de/~thf11669/publications/theis03nonlineargeo_ICA03.pdf.

H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002. URL <http://www.cis.hut.fi/harri/papers/ValpolaNC02.pdf>.

Concentration Bounds for Unigram Language Models

Evgeny Drukh

Yishay Mansour

School of Computer Science

Tel Aviv University

Tel Aviv, 69978, Israel

DRUKH@POST.TAU.AC.IL

MANSOUR@POST.TAU.AC.IL

Editor: John Lafferty

Abstract

We show several high-probability concentration bounds for learning unigram language models. One interesting quantity is the probability of all words appearing exactly k times in a sample of size m . A standard estimator for this quantity is the Good-Turing estimator. The existing analysis on its error shows a high-probability bound of approximately $O\left(\frac{k}{\sqrt{m}}\right)$. We improve its dependency on k to $O\left(\frac{\sqrt[4]{k}}{\sqrt{m}} + \frac{k}{m}\right)$. We also analyze the empirical frequencies estimator, showing that with high probability its error is bounded by approximately $O\left(\frac{1}{k} + \frac{\sqrt{k}}{m}\right)$. We derive a combined estimator, which has an error of approximately $O\left(m^{-\frac{2}{5}}\right)$, for any k .

A standard measure for the quality of a learning algorithm is its expected per-word log-loss. The leave-one-out method can be used for estimating the log-loss of the unigram model. We show that its error has a high-probability bound of approximately $O\left(\frac{1}{\sqrt{m}}\right)$, for any underlying distribution.

We also bound the log-loss a priori, as a function of various parameters of the distribution.

Keywords: Good-Turing estimators, logarithmic loss, leave-one-out estimation, Chernoff bounds

1. Introduction and Overview

Natural language processing (NLP) has developed rapidly over the last decades. It has a wide range of applications, including speech recognition, optical character recognition, text categorization and many more. The theoretical analysis has also advanced significantly, though many fundamental questions remain unanswered. One clear challenge, both practical and theoretical, concerns deriving stochastic models for natural languages.

Consider a simple language model, where the distribution of each word in the text is assumed to be independent. Even for such a simplistic model, fundamental questions relating sample size to the learning accuracy are already challenging. This is mainly due to the fact that the sample size is almost always insufficient, regardless of how large it is.

To demonstrate this phenomena, consider the following example. We would like to estimate the distribution of first names in the university. For that, we are given the names list of a graduate seminar: Alice, Bob, Charlie, Dan, Eve, Frank, two Georges, and two Henriens. How can we use this sample to estimate the distribution of students' first names? An empirical frequency estimator would

assign Alice the probability of 0.1, since there is one Alice in the list of 10 names, while George, appearing twice, would get estimation of 0.2. Unfortunately, unseen names, such as Michael, will get an estimation of 0. Clearly, in this simple example the empirical frequencies are unlikely to estimate well the desired distribution.

In general, the empirical frequencies estimate well the probabilities of popular names, but are rather inaccurate for rare names. Is there a sample size, which assures us that all the names (or most of them) will appear enough times to allow accurate probabilities estimation? The distribution of first names can be conjectured to follow the Zipf’s law. In such distributions, there will be a significant fraction of rare items, as well as a considerable number of non-appearing items, in any sample of reasonable size. The same holds for the language unigram models, which try to estimate the distribution of single words. As it has been observed empirically on many occasions (Chen, 1996; Curran and Osborne, 2002), there are always many rare words and a considerable number of unseen words, regardless of the sample size. Given this observation, a fundamental issue is to estimate the distribution the best way possible.

1.1 Good-Turing Estimators

An important quantity, given a sample, is the probability mass of unseen words (also called “the missing mass”). Several methods exist for smoothing the probability and assigning probability mass to unseen items. The almost standard method for estimating the missing probability mass is the Good-Turing estimator. It estimates the missing mass as the total number of unique items, divided by the sample size. In the names example above, the Good-Turing missing mass estimator is equal 0.6, meaning that the list of the class names does not reflect the true distribution, to put it mildly. The Good-Turing estimator can be extended for higher orders, that is, estimating the probability of all names appearing exactly k times. Such estimators can also be used for estimating the probability of individual words.

The Good-Turing estimators dates back to World War II, and were published first in 1953 (Good, 1953, 2000). It has been extensively used in language modeling applications since then (Katz, 1987; Church and Gale, 1991; Chen, 1996; Chen and Goodman, 1998). However, their theoretical convergence rate in various models has been studied only in the recent years (McAllester and Schapire, 2000, 2001; Kutin, 2002; McAllester and Ortiz, 2003; Orlitsky et al., 2003). For estimation of the probability of all words appearing exactly k times in a sample of size m , McAllester and Schapire (2000) derive a high probability bound on Good-Turing estimator error of approximately $O\left(\frac{k}{\sqrt{m}}\right)$.

One of our main results improves the dependency on k of this bound to approximately $O\left(\frac{\sqrt[4]{k}}{\sqrt{m}} + \frac{k}{m}\right)$. We also show that the empirical frequencies estimator has an error of approximately $O\left(\frac{1}{k} + \frac{\sqrt{k}}{m}\right)$, for large values of k . Based on the two estimators, we derive a combined estimator with an error of approximately $O\left(m^{-\frac{2}{5}}\right)$, for any k . We also derive a weak lower bound of $\Omega\left(\frac{\sqrt[4]{k}}{\sqrt{m}}\right)$ for an error of any estimator based on an independent sample.

Our results give theoretical justification for using the Good-Turing estimator for small values of k , and the empirical frequencies estimator for large values of k . Though in most applications the Good-Turing estimator is used for very small values of k , for example $k \leq 5$, as by Katz (1987) or Chen (1996), we show that it is fairly accurate in a much wider range.

1.2 Logarithmic Loss

The Good-Turing estimators are used to approximate the probability mass of all the words with a certain frequency. For many applications, estimating this probability mass is not the main optimization criteria. Instead, a certain distance measure between the true and the estimated distributions needs to be minimized.

The most popular distance measure used in NLP applications is the *Kullback-Leibler (KL) divergence*. For a true distribution $P = \{p_x\}$, and an estimated distribution $Q = \{q_x\}$, both over some set X , this measure is defined as $\sum_x p_x \ln \frac{p_x}{q_x}$. An equivalent measure, up to the entropy of P , is the *logarithmic loss (log-loss)*, which equals $\sum_x p_x \ln \frac{1}{q_x}$.

Many NLP applications use the value of *log-loss* to evaluate the quality of the estimated distribution. However, the *log-loss* cannot be directly calculated, since it depends on the underlying distribution, which is unknown. Therefore, estimating *log-loss* using the sample is important, although the sample cannot be independently used for both estimating the distribution and testing it. The *hold-out* estimation splits the sample into two parts: training and testing. The training part is used for learning the distribution, whereas the testing sample is used for evaluating the average per-word log-loss. The main disadvantage of this method is the fact that it uses only part of the available information for learning, whereas in practice one would like to use all the sample.

A widely used general estimation method is called *leave-one-out*. Basically, it performs averaging all the possible estimations, where a single item is chosen for testing, and the rest are used for training. This procedure has an advantage of using the entire sample, and in addition it is rather simple and usually can be easily implemented. The existing theoretical analysis of the *leave-one-out* method (Holden, 1996; Kearns and Ron, 1999) shows general high probability concentration bounds for the generalization error. However, these techniques are not applicable in our setting.

We show that the *leave-one-out* estimation error for the *log-loss* is approximately $O\left(\frac{1}{\sqrt{m}}\right)$, for any underlying distribution and a general family of learning algorithms. It gives a theoretical justification for effective use of *leave-one-out* estimation for the *log-loss*.

We also analyze the concentration of the *log-loss* itself, not based of an empirical measure. We address the characteristics of the underlying distribution affecting the *log-loss*. We find such a characteristic, defining a tight bound for the *log-loss* value.

1.3 Model and Semantics

We denote the set of all words as V , and $N = |V|$. Let P be a distribution over V , where p_w is the probability of a word $w \in V$. Given a sample S of size m , drawn i.i.d. using P , we denote the number of appearances of a word w in S as c_w^S , or simply c_w , when a sample S is clear from the context.¹ We define $S_k = \{w \in V : c_w^S = k\}$, and $n_k = |S_k|$.

For a claim Φ regarding a sample S , we write $\forall^\delta S \Phi[S]$ for $P(\Phi[S]) \geq 1 - \delta$. For some error bound function $f(\cdot)$, which holds with probability $1 - \delta$, we write $\tilde{O}(f(\cdot))$ for $O(f(\cdot) (\ln \frac{m}{\delta})^c)$, where $c > 0$ is some constant.

1.4 Paper Organization

Section 2 shows several standard concentration inequalities, together with their technical applications regarding the maximum-likelihood approximation. Section 3 shows the error bounds for the

1. Unless mentioned otherwise, all further sample-dependent definitions depend on the sample S .

k -hitting mass estimation. Section 4 bounds the error for the leave-one-out estimation of the logarithmic loss. Section 5 shows the bounds for the a priori logarithmic loss. Appendix A includes the technical proofs.

2. Concentration Inequalities

In this section we state several standard Chernoff-style concentration inequalities. We also show some of their corollaries regarding the maximum-likelihood approximation of p_w by $\hat{p}_w = \frac{c_w}{m}$.

Lemma 1 (Hoeffding, 1963) *Let $Y = Y_1, \dots, Y_n$ be a set of n independent random variables, such that $Y_i \in [b_i, b_i + d_i]$. Then, for any $\varepsilon > 0$,*

$$P\left(\left|\sum_i Y_i - E\left[\sum_i Y_i\right]\right| > \varepsilon\right) \leq 2 \exp\left(-\frac{2\varepsilon^2}{\sum_i d_i^2}\right).$$

The next lemma is a variant of an extension of Hoeffding’s inequality, by McDiarmid (1989).

Lemma 2 *Let $Y = Y_1, \dots, Y_n$ be a set of n independent random variables, and $f(Y)$ such that any change of Y_i value changes $f(Y)$ by at most d_i , that is*

$$\sup_{\forall j \neq i, Y_j = Y'_j} (|f(Y) - f(Y')|) \leq d_i.$$

Let $d = \max_i d_i$. Then,

$$\forall \delta Y : |f(Y) - E[f(Y)]| \leq d \sqrt{\frac{n \ln \frac{2}{\delta}}{2}}.$$

Lemma 3 (Angluin and Valiant, 1979) *Let $Y = Y_1, \dots, Y_n$ be a set of n independent random variables, where $Y_i \in [0, B]$. Let $\mu = E[\sum_i Y_i]$. Then, for any $\varepsilon > 0$,*

$$\begin{aligned} P\left(\sum_i Y_i < \mu - \varepsilon\right) &\leq \exp\left(-\frac{\varepsilon^2}{2\mu B}\right), \\ P\left(\sum_i Y_i > \mu + \varepsilon\right) &\leq \exp\left(-\frac{\varepsilon^2}{(2\mu + \varepsilon)B}\right). \end{aligned}$$

Definition 4 (Dubhashi and Ranjan, 1998) *A set of random variables Y_1, \dots, Y_n is called “negatively associated”, if it satisfies for any two disjoint subsets I and J of $\{1, \dots, n\}$, and any two non-decreasing, or any two non-increasing, functions f from $\mathbb{R}^{|I|}$ to \mathbb{R} and g from $\mathbb{R}^{|J|}$ to \mathbb{R} :*

$$E[f(Y_i : i \in I)g(Y_j : j \in J)] \leq E[f(Y_i : i \in I)]E[g(Y_j : j \in J)].$$

The next lemma is based on the *negative association* analysis. It follows directly from Theorem 14 and Proposition 7 of Dubhashi and Ranjan (1998).

Lemma 5 For any set of N non-decreasing, or N non-increasing functions $\{f_w : w \in V\}$, any Chernoff-style bound on $\sum_{w \in V} f_w(c_w)$, pretending that c_w are independent, is valid. In particular, Lemmas 1 and 2 apply for $\{Y_1, \dots, Y_n\} = \{f_w(c_w) : w \in V\}$.

The next lemma shows an explicit upper bound on the binomial distribution probability.²

Lemma 6 Let $X \sim \text{Bin}(n, p)$ be a sum of n i.i.d. Bernoulli random variables with $p \in (0, 1)$. Let $\mu = E[X] = np$. For $x \in (0, n]$, there exists some function $T_x = \exp\left(\frac{1}{12x} + O\left(\frac{1}{x^2}\right)\right)$, such that $\forall k \in \{0, \dots, n\}$, we have $P(X = k) \leq \frac{1}{\sqrt{2\pi\mu(1-p)}} \frac{T_n}{T_\mu T_{n-\mu}}$. For integral values of μ , the equality is achieved at $k = \mu$. (Note that for $x \geq 1$, we have $T_x = \Theta(1)$.)

The next lemma deals with the number of successes in independent trials.

Lemma 7 (Hoeffding, 1956) Let $Y_1, \dots, Y_n \in \{0, 1\}$ be a sequence of independent trials, with $p_i = E[Y_i]$. Let $X = \sum_i Y_i$ be the number of successes, and $p = \frac{1}{n} \sum_i p_i$ be the average trial success probability. For any integers b and c such that $0 \leq b \leq np \leq c \leq n$, we have

$$\sum_{k=b}^c \binom{n}{k} p^k (1-p)^{n-k} \leq P(b \leq X \leq c) \leq 1.$$

Using the above lemma, the next lemma shows a general concentration bound for a sum of arbitrary real-valued functions of a multinomial distribution components. We show that with a small penalty, any Chernoff-style bound pretending the components being independent is valid.³ We recall that c_w^S , or equivalently c_w , is the number of appearances of the word w in a sample S of size m .

Lemma 8 Let $\{c'_w \sim \text{Bin}(m, p_w) : w \in V\}$ be independent binomial random variables. Let $\{f_w(x) : w \in V\}$ be a set of real valued functions. Let $F = \sum_w f_w(c_w)$ and $F' = \sum_w f_w(c'_w)$. For any $\epsilon > 0$,

$$P(|F - E[F]| > \epsilon) \leq 3\sqrt{m} P(|F' - E[F']| > \epsilon).$$

The following lemmas provide concentration bounds for maximum-likelihood estimation of p_w by $\hat{p}_w = \frac{c_w}{m}$. The first lemma shows that words with “high” probability have a “high” count in the sample.

Lemma 9 Let $\delta > 0$, and $\lambda \geq 3$. We have $\forall^\delta S$:

$$\begin{aligned} \forall w \in V, \text{ s.t. } \quad mp_w \geq 3 \ln \frac{2m}{\delta}, \quad |mp_w - c_w| &\leq \sqrt{3mp_w \ln \frac{2m}{\delta}}; \\ \forall w \in V, \text{ s.t. } \quad mp_w > \lambda \ln \frac{2m}{\delta}, \quad c_w &> \left(1 - \sqrt{\frac{3}{\lambda}}\right) mp_w. \end{aligned}$$

2. Its proof is based on Stirling approximation directly, though local limit theorems could be used. This form of bound is needed for the proof of Theorem 30.

3. The *negative association* analysis (Lemma 5) shows that a sum of monotone functions of multinomial distribution components must obey Chernoff-style bounds pretending that the components are independent. In some sense, our result extends this notion, since it does not require the functions to be monotone.

The second lemma shows that words with “low” probability have a “low” count in the sample.

Lemma 10 *Let $\delta \in (0, 1)$, and $m > 1$. Then, $\forall^\delta S$: $\forall w \in V$ such that $mp_w \leq 3\ln \frac{m}{\delta}$, we have $c_w \leq 6\ln \frac{m}{\delta}$.*

The following lemma derives the bound as a function of the count in the sample (and not as a function of the unknown probability).

Lemma 11 *Let $\delta > 0$. Then, $\forall^\delta S$:*

$$\forall w \in V, \text{ s.t. } c_w > 18\ln \frac{4m}{\delta}, \quad |mp_w - c_w| \leq \sqrt{6c_w \ln \frac{4m}{\delta}}.$$

The following is a general concentration bound.

Lemma 12 *For any $\delta > 0$, and any word $w \in V$, we have*

$$\forall^\delta S, \quad \left| \frac{c_w}{m} - p_w \right| < \sqrt{\frac{3\ln \frac{2}{\delta}}{m}}.$$

The following lemma bounds the probability of words that do not appear in the sample.

Lemma 13 *Let $\delta > 0$. Then, $\forall^\delta S$:*

$$\forall w \notin S, \quad mp_w < \ln \frac{m}{\delta}.$$

3. K-Hitting Mass Estimation

In this section our goal is to estimate the probability of the set of words appearing exactly k times in the sample, which we call “the k -hitting mass”. We analyze the Good-Turing estimator, the empirical frequencies estimator, and a combined estimator.

Definition 14 *We define the k -hitting mass M_k , its empirical frequencies estimator \hat{M}_k , and its Good-Turing estimator G_k as⁴*

$$M_k = \sum_{w \in S_k} p_w \quad \hat{M}_k = \binom{k}{m} n_k \quad G_k = \binom{k+1}{m-k} n_{k+1}.$$

The outline of this section is as follows. Definition 16 slightly redefines the k -hitting mass and its estimators. Lemma 17 shows that this redefinition has a negligible influence. Then, we analyze the estimation errors using the concentration inequalities from Section 2.

Lemmas 20 and 21 bound the expectation of the Good-Turing estimator error, following McAllester and Schapire (2000). Lemma 23 bounds the deviation of the error, using the negative association analysis. A tighter bound, based on Lemma 8, is achieved at Theorem 25. Theorem 26 analyzes the error of the empirical frequencies estimator. Theorem 29 refers to the combined estimator. Finally, Theorem 30 shows a weak lower bound for the k -hitting mass estimation.

4. The Good-Turing estimator is usually defined as $\binom{k+1}{m} n_{k+1}$. The two definitions are almost identical for small values of k , as their quotient equals $1 - \frac{k}{m}$. Following McAllester and Schapire (2000), our definition makes the calculations slightly simpler.

Definition 15 For any $w \in V$ and $i \in \{0, \dots, m\}$, we define $X_{w,i}$ as a random variable equal 1 if $c_w = i$, and 0 otherwise.

The following definition concentrates on words whose frequencies are close to their probabilities.

Definition 16 Let $\alpha > 0$ and $k > 3\alpha^2$. We define $I_{k,\alpha} = \left[\frac{k-\alpha\sqrt{k}}{m}, \frac{k+1+\alpha\sqrt{k+1}}{m} \right]$, and $V_{k,\alpha} = \{w \in V : p_w \in I_{k,\alpha}\}$. We define:

$$\begin{aligned} M_{k,\alpha} &= \sum_{w \in S_k \cap V_{k,\alpha}} p_w = \sum_{w \in V_{k,\alpha}} p_w X_{w,k}, \\ G_{k,\alpha} &= \frac{k+1}{m-k} |S_{k+1} \cap V_{k,\alpha}| = \frac{k+1}{m-k} \sum_{w \in V_{k,\alpha}} X_{w,k+1}, \\ \hat{M}_{k,\alpha} &= \frac{k}{m} |S_k \cap V_{k,\alpha}| = \frac{k}{m} \sum_{w \in V_{k,\alpha}} X_{w,k}. \end{aligned}$$

By Lemma 11, for large values of k the redefinition coincides with the original definition with high probability:

Lemma 17 For $\delta > 0$, let $\alpha = \sqrt{6 \ln \frac{4m}{\delta}}$. For $k > 18 \ln \frac{4m}{\delta}$, we have $\forall^\delta S$: $M_k = M_{k,\alpha}$, $G_k = G_{k,\alpha}$, and $\hat{M}_k = \hat{M}_{k,\alpha}$.

Proof By Lemma 11, we have

$$\forall^\delta S, \quad \forall w : c_w > 18 \ln \frac{4m}{\delta}, \quad |mp_w - c_w| \leq \sqrt{6c_w \ln \frac{4m}{\delta}} = \alpha \sqrt{c_w}.$$

This means that any word w with $c_w = k$ has

$$\frac{k - \alpha\sqrt{k}}{m} \leq p_w \leq \frac{k + \alpha\sqrt{k}}{m} < \frac{k + 1 + \alpha\sqrt{k+1}}{m}.$$

Therefore $w \in V_{k,\alpha}$, completing the proof for M_k and \hat{M}_k . Since $\alpha < \sqrt{k}$, any word w with $c_w = k+1$ has

$$\frac{k - \alpha\sqrt{k}}{m} < \frac{k + 1 - \alpha\sqrt{k+1}}{m} \leq p_w \leq \frac{k + 1 + \alpha\sqrt{k+1}}{m},$$

which yields $w \in V_{k,\alpha}$, completing the proof for G_k . ■

Since the minimal probability of a word in $V_{k,\alpha}$ is $\Omega\left(\frac{k}{m}\right)$, we derive:

Lemma 18 Let $\alpha > 0$ and $k > 3\alpha^2$. Then, $|V_{k,\alpha}| = O\left(\frac{m}{k}\right)$.

Proof We have $\alpha < \frac{\sqrt{k}}{\sqrt{3}}$. Any word $w \in V_{k,\alpha}$ has $p_w \geq \frac{k-\alpha\sqrt{k}}{m} > \frac{k}{m} \left(1 - \frac{1}{\sqrt{3}}\right)$. Therefore,

$$|V_{k,\alpha}| < \frac{m}{k} \frac{1}{1 - \frac{1}{\sqrt{3}}} = O\left(\frac{m}{k}\right),$$

which completes the proof. ■

Using Lemma 6, we derive:

Lemma 19 *Let $\alpha > 0$ and $3\alpha^2 < k \leq \frac{m}{2}$. Let $w \in V_{k,\alpha}$. Then, $E[X_{w,k}] = P(c_w = k) = O\left(\frac{1}{\sqrt{k}}\right)$.*

Proof Since $c_w \sim \text{Bin}(m, p_w)$ is a binomial random variable, we use Lemma 6:

$$E[X_{w,k}] = P(c_w = k) \leq \frac{1}{\sqrt{2\pi m p_w (1-p_w)}} \frac{T_m}{T_{mp_w} T_{m(1-p_w)}}.$$

For $w \in V_{k,\alpha}$, we have $mp_w = \Omega(k)$, which implies $\frac{T_m}{T_{mp_w} T_{m(1-p_w)}} = O(1)$. Since $p_w \in I_{k,\alpha}$ and $3\alpha^2 < k \leq \frac{m}{2}$, we have

$$\begin{aligned} \frac{1}{\sqrt{2\pi m p_w (1-p_w)}} &\leq \frac{1}{\sqrt{2\pi \left(k - \alpha\sqrt{k}\right) \left(1 - \left(\frac{k+1+\alpha\sqrt{k+1}}{m}\right)\right)}} \\ &< \frac{1}{\sqrt{2\pi k \left(1 - \frac{1}{\sqrt{3}}\right) \left(1 - \frac{k+1}{m} \left(1 + \frac{1}{\sqrt{3}}\right)\right)}} \\ &< \frac{1}{\sqrt{2\pi k \left(1 - \frac{1}{\sqrt{3}}\right) \left(1 - \left(\frac{1}{2} + \frac{1}{m}\right) \left(1 + \frac{1}{\sqrt{3}}\right)\right)}} \\ &= O\left(\frac{1}{\sqrt{k}}\right), \end{aligned}$$

which completes the proof. ■

3.1 Good-Turing Estimator

The following lemma, directly based on the definition of the binomial distribution, was shown in Theorem 1 of McAllester and Schapire (2000).

Lemma 20 *For any $k < m$, and $w \in V$, we have*

$$p_w P(c_w = k) = \frac{k+1}{m-k} P(c_w = k+1)(1-p_w).$$

The following lemma bounds the expectations of the redefined k -hitting mass, its Good-Turing estimator, and their difference.

Lemma 21 Let $\alpha > 0$ and $3\alpha^2 < k < \frac{m}{2}$. We have $E[M_{k,\alpha}] = O\left(\frac{1}{\sqrt{k}}\right)$, $E[G_{k,\alpha}] = O\left(\frac{1}{\sqrt{k}}\right)$, and $|E[G_{k,\alpha}] - E[M_{k,\alpha}]| = O\left(\frac{\sqrt{k}}{m}\right)$.

Lemma 22 Let $\delta > 0$, $k \in \{1, \dots, m\}$. Let $U \subseteq V$, such that $|U| = O\left(\frac{m}{k}\right)$. Let $\{b_w : w \in U\}$, such that $\forall w \in U, b_w \geq 0$ and $\max_{w \in U} b_w = O\left(\frac{k}{m}\right)$. Let $X_k = \sum_{w \in U} b_w X_{w,k}$. We have $\forall^\delta S$:

$$|X_k - E[X_k]| = O\left(\sqrt{\frac{k \ln \frac{1}{\delta}}{m}}\right).$$

Proof We define $Y_{w,k} = \sum_{i \leq k} X_{w,i}$ be random variable indicating $c_w \leq k$ and $Z_{w,k} = \sum_{i < k} X_{w,i} = Y_{w,k} - X_{w,k}$ be random variable indicating $c_w < k$. Let $Y_k = \sum_{w \in U} b_w Y_{w,k}$ and $Z_k = \sum_{w \in U} b_w Z_{w,k}$. We have

$$X_k = \sum_{w \in U} b_w X_{w,k} = \sum_{w \in U} b_w [Y_{w,k} - Z_{w,k}] = Y_k - Z_k.$$

Both Y_k and Z_k , can be bounded using the Hoeffding inequality. Since $\{b_w Y_{w,k}\}$ and $\{b_w Z_{w,k}\}$ are monotone with respect to $\{c_w\}$, Lemma 5 applies for them. This means that the concentration of their sum is at least as tight as if they were independent. Recalling that $|U| = O\left(\frac{m}{k}\right)$ and $\max_{w \in U} b_w = O\left(\frac{k}{m}\right)$, and using Lemma 2 for Y_k and Z_k , we have

$$\begin{aligned} \forall^{\frac{\delta}{2}} S, \quad |Y_k - E[Y_k]| &= O\left(\frac{k}{m} \sqrt{\frac{m}{k} \ln \frac{1}{\delta}}\right), \\ \forall^{\frac{\delta}{2}} S, \quad |Z_k - E[Z_k]| &= O\left(\frac{k}{m} \sqrt{\frac{m}{k} \ln \frac{1}{\delta}}\right). \end{aligned}$$

Therefore,

$$\begin{aligned} |X_k - E[X_k]| &= |Y_k - Z_k - E[Y_k - Z_k]| \\ &\leq |Y_k - E[Y_k]| + |Z_k - E[Z_k]| = O\left(\sqrt{\frac{k \ln \frac{1}{\delta}}{m}}\right), \end{aligned}$$

which completes the proof. ■

Using the *negative association* notion, we can show a preliminary bound for Good-Turing estimation error:

Lemma 23 For $\delta > 0$ and $18 \ln \frac{8m}{\delta} < k < \frac{m}{2}$, we have $\forall^\delta S$:

$$|G_k - M_k| = O\left(\sqrt{\frac{k \ln \frac{1}{\delta}}{m}}\right).$$

Proof Let $\alpha = \sqrt{6 \ln \frac{8m}{\delta}}$. By Lemma 17, we have

$$\forall^{\frac{\delta}{2}} S, \quad G_k = G_{k,\alpha} \wedge M_k = M_{k,\alpha}. \quad (1)$$

By Lemma 21,

$$|E[G_k - M_k]| = |E[G_{k,\alpha} - M_{k,\alpha}]| = O\left(\frac{\sqrt{k}}{m}\right). \quad (2)$$

By Definition 16, $M_{k,\alpha} = \sum_{w \in V_{k,\alpha}} p_w X_{w,k}$ and $G_{k,\alpha} = \sum_{w \in V_{k,\alpha}} \left(\frac{k+1}{m-k}\right) X_{w,k+1}$. By Lemma 18, we have $|V_{k,\alpha}| = O\left(\frac{m}{k}\right)$. Therefore, using Lemma 22 with k for $M_{k,\alpha}$, and with $k+1$ for $G_{k,\alpha}$, we have

$$\forall^{\frac{\delta}{4}} S, \quad |M_{k,\alpha} - E[M_{k,\alpha}]| = O\left(\sqrt{\frac{k \ln \frac{1}{\delta}}{m}}\right), \quad (3)$$

$$\forall^{\frac{\delta}{4}} S, \quad |G_{k,\alpha} - E[G_{k,\alpha}]| = O\left(\sqrt{\frac{k \ln \frac{1}{\delta}}{m}}\right). \quad (4)$$

Combining Equations (1), (2), (3), and (4), we have $\forall^{\delta} S$:

$$\begin{aligned} |G_k - M_k| &= |G_{k,\alpha} - M_{k,\alpha}| \\ &\leq |G_{k,\alpha} - E[G_{k,\alpha}]| + |M_{k,\alpha} - E[M_{k,\alpha}]| + |E[G_{k,\alpha}] - E[M_{k,\alpha}]| \\ &= O\left(\sqrt{\frac{k \ln \frac{1}{\delta}}{m}}\right) + O\left(\frac{\sqrt{k}}{m}\right) = O\left(\sqrt{\frac{k \ln \frac{1}{\delta}}{m}}\right), \end{aligned}$$

which completes the proof. ■

Lemma 24 Let $\delta > 0$, $k > 0$. Let $U \subseteq V$. Let $\{b_w : w \in U\}$ be a set of weights, such that $b_w \in [0, B]$. Let $X_k = \sum_{w \in U} b_w X_{w,k}$, and $\mu = E[X_k]$. We have

$$\forall^{\delta} S, \quad |X_k - \mu| \leq \max \left\{ \sqrt{4B\mu \ln \left(\frac{6\sqrt{m}}{\delta}\right)}, 2B \ln \left(\frac{6\sqrt{m}}{\delta}\right) \right\}.$$

Proof By Lemma 8, combined with Lemma 3, we have

$$\begin{aligned} P(|X_k - \mu| > \varepsilon) &\leq 6\sqrt{m} \exp\left(-\frac{\varepsilon^2}{B(2\mu + \varepsilon)}\right) \\ &\leq \max \left\{ 6\sqrt{m} \exp\left(-\frac{\varepsilon^2}{4B\mu}\right), 6\sqrt{m} \exp\left(-\frac{\varepsilon}{2B}\right) \right\}, \quad (5) \end{aligned}$$

where Equation (5) follows by considering $\varepsilon \leq 2\mu$ and $\varepsilon > 2\mu$ separately. The lemma follows substituting $\varepsilon = \max \left\{ \sqrt{4B\mu \ln \left(\frac{6\sqrt{m}}{\delta} \right)}, 2B \ln \left(\frac{6\sqrt{m}}{\delta} \right) \right\}$. ■

We now derive the concentration bound on the error of the Good-Turing estimator.

Theorem 25 For $\delta > 0$ and $18 \ln \frac{8m}{\delta} < k < \frac{m}{2}$, we have $\forall^\delta S$:

$$|G_k - M_k| = O \left(\sqrt{\frac{\sqrt{k} \ln \frac{m}{\delta}}{m}} + \frac{k \ln \frac{m}{\delta}}{m} \right).$$

Proof Let $\alpha = \sqrt{6 \ln \frac{8m}{\delta}}$. Using Lemma 17, we have $\forall^{\frac{\delta}{2}} S$: $G_k = G_{k,\alpha}$, and $M_k = M_{k,\alpha}$. Recall that $M_{k,\alpha} = \sum_{w \in V_{k,\alpha}} p_w X_{w,k}$ and $G_{k,\alpha} = \sum_{w \in V_{k,\alpha}} \frac{k+1}{m-k} X_{w,k+1}$. Both $M_{k,\alpha}$ and $G_{k,\alpha}$ are linear combinations of $X_{w,k}$ and $X_{w,k+1}$, respectively, where the coefficients' magnitude is $O\left(\frac{k}{m}\right)$, and the expectation, by Lemma 21, is $O\left(\frac{1}{\sqrt{k}}\right)$. By Lemma 24, we have

$$\forall^{\frac{\delta}{4}} S, \quad |M_{k,\alpha} - E[M_{k,\alpha}]| = O \left(\sqrt{\frac{\sqrt{k} \ln \frac{m}{\delta}}{m}} + \frac{k \ln \frac{m}{\delta}}{m} \right), \tag{6}$$

$$\forall^{\frac{\delta}{4}} S, \quad |G_{k,\alpha} - E[G_{k,\alpha}]| = O \left(\sqrt{\frac{\sqrt{k} \ln \frac{m}{\delta}}{m}} + \frac{k \ln \frac{m}{\delta}}{m} \right). \tag{7}$$

Combining Equations (6), (7), and Lemma 21, we have $\forall^\delta S$:

$$\begin{aligned} |G_k - M_k| &= |G_{k,\alpha} - M_{k,\alpha}| \\ &\leq |G_{k,\alpha} - E[G_{k,\alpha}]| + |M_{k,\alpha} - E[M_{k,\alpha}]| + |E[G_{k,\alpha}] - E[M_{k,\alpha}]| \\ &= O \left(\sqrt{\frac{\sqrt{k} \ln \frac{m}{\delta}}{m}} + \frac{k \ln \frac{m}{\delta}}{m} + \frac{\sqrt{k}}{m} \right) = O \left(\sqrt{\frac{\sqrt{k} \ln \frac{m}{\delta}}{m}} + \frac{k \ln \frac{m}{\delta}}{m} \right), \end{aligned}$$

which completes the proof. ■

3.2 Empirical Frequencies Estimator

In this section we bound the error of the empirical frequencies estimator \hat{M}_k .

Theorem 26 For $\delta > 0$ and $18 \ln \frac{8m}{\delta} < k < \frac{m}{2}$, we have

$$\forall^\delta S, \quad |M_k - \hat{M}_k| = O \left(\frac{\sqrt{k} \left(\ln \frac{m}{\delta} \right)^{\frac{3}{2}}}{m} + \frac{\sqrt{\ln \frac{m}{\delta}}}{k} \right).$$

Proof Let $\alpha = \sqrt{6 \ln \frac{8m}{\delta}}$. By Lemma 17, we have $\forall^{\frac{\delta}{2}} S$: $\hat{M}_k = \hat{M}_{k,\alpha}$, and $M_k = M_{k,\alpha}$. Let $V_{k,\alpha}^- = \{w \in V_{k,\alpha} : p_w < \frac{k}{m}\}$, and $V_{k,\alpha}^+ = \{w \in V_{k,\alpha} : p_w > \frac{k}{m}\}$. Let

$$X_- = \sum_{w \in V_{k,\alpha}^-} \left(\frac{k}{m} - p_w \right) X_{w,k}, \quad X_+ = \sum_{w \in V_{k,\alpha}^+} \left(p_w - \frac{k}{m} \right) X_{w,k},$$

and let $X_{\text{?}}$ specify either X_- or X_+ . By the definition, for $w \in V_{k,\alpha}$ we have $|\frac{k}{m} - p_w| = O\left(\frac{\alpha\sqrt{k}}{m}\right)$. By Lemma 18, $|V_{k,\alpha}| = O\left(\frac{m}{k}\right)$. By Lemma 19, for $w \in V_{k,\alpha}$ we have $E[X_{w,k}] = O\left(\frac{1}{\sqrt{k}}\right)$. Therefore,

$$|E[X_{\text{?}}]| \leq \sum_{w \in V_{k,\alpha}} \left| \frac{k}{m} - p_w \right| E[X_{w,k}] = O\left(\frac{m \alpha \sqrt{k}}{k} \frac{1}{m} \frac{1}{\sqrt{k}}\right) = O\left(\frac{\alpha}{k}\right). \quad (8)$$

Both X_- and X_+ are linear combinations of $X_{w,k}$, where the coefficients are $O\left(\frac{\alpha\sqrt{k}}{m}\right)$ and the expectation is $O\left(\frac{\alpha}{k}\right)$. Therefore, by Lemma 24, we have

$$\forall^{\frac{\delta}{4}} S: \quad |X_{\text{?}} - E[X_{\text{?}}]| = O\left(\sqrt{\frac{\alpha^4}{m\sqrt{k}} + \frac{\alpha^3\sqrt{k}}{m}}\right). \quad (9)$$

By the definition of X_- and X_+ , $M_{k,\alpha} - \hat{M}_{k,\alpha} = X_+ - X_-$. Combining Equations (8) and (9), we have $\forall^{\frac{\delta}{8}} S$:

$$\begin{aligned} |M_k - \hat{M}_k| &= |M_{k,\alpha} - \hat{M}_{k,\alpha}| = |X_+ - X_-| \\ &\leq |X_+ - E[X_+]| + |E[X_+]| + |X_- - E[X_-]| + |E[X_-]| \\ &= O\left(\sqrt{\frac{\alpha^4}{m\sqrt{k}} + \frac{\alpha^3\sqrt{k}}{m}} + \frac{\alpha}{k}\right) = O\left(\frac{\sqrt{k} \left(\ln \frac{m}{\delta}\right)^{\frac{3}{2}}}{m} + \frac{\sqrt{\ln \frac{m}{\delta}}}{k}\right), \end{aligned}$$

since $\sqrt{ab} = O(a+b)$, and we use $a = \frac{\alpha^3\sqrt{k}}{m}$ and $b = \frac{\alpha}{k}$. ■

3.3 Combined Estimator

In this section we combine the Good-Turing estimator with the empirical frequencies to derive a combined estimator, which is uniformly accurate for all values of k .

Definition 27 We define \tilde{M}_k , a combined estimator for M_k , by

$$\tilde{M}_k = \begin{cases} G_k & k \leq m^{\frac{2}{5}} \\ \hat{M}_k & k > m^{\frac{2}{5}}. \end{cases}$$

Lemma 28 (McAllester and Schapire, 2000) *Let $k \in \{0, \dots, m\}$. For any $\delta > 0$, we have*

$$\forall \delta > 0: \quad |G_k - M_k| = O\left(\sqrt{\frac{\ln \frac{1}{\delta}}{m}} \left(k + \ln \frac{m}{\delta}\right)\right).$$

The following theorem shows that \tilde{M}_k has an error bounded by $\tilde{O}\left(m^{-\frac{2}{5}}\right)$, for any k . For small k , we use Lemma 28. Theorem 25 is used for $18 \ln \frac{8m}{\delta} < k \leq m^{\frac{2}{5}}$. Theorem 26 is used for $m^{\frac{2}{5}} < k < \frac{m}{2}$. The complete proof also handles $k \geq \frac{m}{2}$. The theorem refers to \tilde{M}_k as a probability estimator, and does not show that it is a probability distribution by itself.

Theorem 29 *Let $\delta > 0$. For any $k \in \{0, \dots, m\}$, we have*

$$\forall \delta > 0, \quad |\tilde{M}_k - M_k| = \tilde{O}\left(m^{-\frac{2}{5}}\right).$$

The following theorem shows a weak lower bound for approximating M_k . It applies to estimating M_k based on a different independent sample. This is a very “weak” notation, since G_k , as well as \tilde{M}_k , are based on the same sample as M_k .

Theorem 30 *Suppose that the vocabulary consists of $\frac{m}{k}$ words distributed uniformly (that is $p_w = \frac{k}{m}$), where $1 \ll k \ll m$. The variance of M_k is $\Theta\left(\frac{\sqrt{k}}{m}\right)$.*

4. Leave-One-Out Estimation of Log-Loss

Many NLP applications use log-loss as the learning performance criteria. Since the log-loss depends on the underlying probability P , its value cannot be explicitly calculated, and must be approximated. The main result of this section, Theorem 32, is an upper bound on the leave-one-out estimation of the log-loss, assuming a general family of learning algorithms.

Given a sample $S = \{s_1, \dots, s_m\}$, the goal of a learning algorithm is to approximate the true probability P by some probability Q . We denote the probability assigned by the learning algorithm to a word w by q_w .

Definition 31 *We assume that any two words with equal sample frequency are assigned equal probabilities in Q , and therefore denote q_w by $q(c_w)$. Let the log-loss of a distribution Q be*

$$L = \sum_{w \in V} p_w \ln \frac{1}{q_w} = \sum_{k \geq 0} M_k \ln \frac{1}{q(k)}.$$

Let the leave-one-out estimation, q'_w , be the probability assigned to w , when one of its instances is removed. We assume that any two words with equal sample frequency are assigned equal leave-one-out probability estimation, and therefore denote q'_w by $q'(c_w)$. We define the leave-one-out estimation of the log-loss as averaging the loss of each sample word, when it is extracted from the sample and pretended to be the test sample:

$$L_{leave-one} = \sum_{w \in V} \frac{c_w}{m} \ln \frac{1}{q'_w} = \sum_{k>0} \frac{kn_k}{m} \ln \frac{1}{q'(k)}.$$

Let $L_w = L(c_w) = \ln \frac{1}{q(c_w)}$, and $L'_w = L'(c_w) = \ln \frac{1}{q'(c_w)}$. Let the maximal loss be

$$L_{max} = \max_k \max \{L(k), L'(k+1)\}.$$

In this section we discuss a family of learning algorithms, that receive the sample as an input. Assuming an accuracy parameter δ , we require the following properties to hold:

1. Starting from a certain number of appearances, the estimation is close to the sample frequency. Specifically, for some $\alpha, \beta \in [0, 1]$,

$$\forall k \geq \ln \left(\frac{4m}{\delta} \right), \quad q(k) = \frac{k - \alpha}{m - \beta}. \tag{10}$$

2. The algorithm is stable when a single word is extracted from the sample:

$$\forall m, \quad 2 \leq k \leq 10 \ln \frac{4m}{\delta}, \quad |L'(k+1) - L(k)| = O\left(\frac{1}{m}\right), \tag{11}$$

$$\forall m, \quad \forall S \text{ s.t. } n_1^S > 0, \quad k \in \{0, 1\}, \quad |L'(k+1) - L(k)| = O\left(\frac{1}{n_1^S}\right). \tag{12}$$

An example of such an algorithm is the following leave-one-out algorithm (we assume that the vocabulary is large enough so that $n_0 + n_1 > 0$):

$$q_w = \begin{cases} \frac{N - n_0 - 1}{(n_0 + n_1)(m - 1)} & c_w \leq 1 \\ \frac{c_w - 1}{m - 1} & c_w \geq 2. \end{cases}$$

Equation (10) is satisfied by $\alpha = \beta = 1$. Equation (11) is satisfied for $k \geq 2$ by $L(k) - L'(k+1) = \ln \left(\frac{m-1}{m-2} \right) = O\left(\frac{1}{m}\right)$. Equation (12) is satisfied for $k \leq 1$:

$$|L'(1) - L(0)| = \left| \ln \left(\frac{N - n_0 - 1}{N - n_0 - 2} \frac{m - 2}{m - 1} \right) \right| = O\left(\frac{1}{N - n_0} + \frac{1}{m}\right) = O\left(\frac{1}{n_1}\right),$$

$$|L'(2) - L(1)| = \left| \ln \left(\frac{n_0 + n_1 + 1}{n_0 + n_1} \frac{m - 2}{m - 1} \right) \right| = O\left(\frac{1}{n_0 + n_1} + \frac{1}{m}\right) = O\left(\frac{1}{n_1}\right).$$

The following is the main theorem of this section. It bounds the deviation between the true loss and the *leave one out* estimate. This bound shows that for a general family of learning algorithms, leave-one-out technique can be effectively used to estimate the logarithmic loss, given the sample only. The estimation error bound decreases roughly in proportion to the square root of the sample size, regardless of the underlying distribution.

Theorem 32 For a learning algorithm satisfying Equations (10), (11), and (12), and $\delta > 0$, we have:

$$\forall \delta S, \quad |L - L_{\text{leave-one}}| = O\left(L_{\max} \sqrt{\frac{(\ln \frac{m}{\delta})^4 \ln \frac{m}{\delta}}{m}}\right).$$

The proof of Theorem 32 bounds the estimation error separately for the high-probability and low-probability words. We use Lemma 20 (McAllester and Schapire, 2000) to bound the estimation error for low-probability words. The expected estimation error for the high-probability words is bounded elementarily using the definition of the binomial distribution (Lemma 33). Finally, we use McDiarmid's inequality (Lemma 2) to bound its deviation.

The next lemma shows that the expectation of the leave-one-out method is a good approximation for the per-word expectation of the logarithmic loss.

Lemma 33 Let $0 \leq \alpha \leq 1$, and $y \geq 1$. Let $B_n \sim \text{Bin}(n, p)$ be a binomial random variable. Let $f_y(x) = \ln(\max(x, y))$. Then,

$$0 \leq E\left[p f_y(B_n - \alpha) - \frac{B_n}{n} f_y(B_n - \alpha - 1)\right] \leq \frac{3p}{n}.$$

Proof For a real valued function F (here $F(x) = f_y(x - \alpha)$), we have:

$$\begin{aligned} E\left[\frac{B_n}{n} F(B_n - 1)\right] &= \sum_{x=0}^n \binom{n}{x} p^x (1-p)^{n-x} \frac{x}{n} F(x-1) \\ &= p \sum_{x=1}^n \binom{n-1}{x-1} p^{x-1} (1-p)^{(n-1)-(x-1)} F(x-1) \\ &= p E[F(B_{n-1})], \end{aligned}$$

where we used $\binom{n}{x} \frac{x}{n} = \binom{n-1}{x-1}$. Since $B_n \sim B_{n-1} + B_1$, we have:

$$\begin{aligned} E\left[p f_y(B_n - \alpha) - \frac{B_n}{n} f_y(B_n - \alpha - 1)\right] &= p(E[f_y(B_{n-1} + B_1 - \alpha)] - E[f_y(B_{n-1} - \alpha)]) \\ &= pE\left[\ln \frac{\max(B_{n-1} + B_1 - \alpha, y)}{\max(B_{n-1} - \alpha, y)}\right] \\ &\leq pE\left[\ln \frac{\max(B_{n-1} - \alpha + B_1, y + B_1)}{\max(B_{n-1} - \alpha, y)}\right] \\ &= pE\left[\ln\left(1 + \frac{B_1}{\max(B_{n-1} - \alpha, y)}\right)\right] \\ &\leq pE\left[\frac{B_1}{\max(B_{n-1} - \alpha, y)}\right]. \end{aligned}$$

Since B_1 and B_{n-1} are independent, we get

$$\begin{aligned}
 pE \left[\frac{B_1}{\max(B_{n-1} - \alpha, y)} \right] &= pE[B_1]E \left[\frac{1}{\max(B_{n-1} - \alpha, y)} \right] \\
 &= p^2 E \left[\frac{1}{\max(B_{n-1} - \alpha, y)} \right] \\
 &= p^2 \sum_{x=0}^{n-1} \binom{n-1}{x} p^x (1-p)^{n-1-x} \frac{1}{\max(x - \alpha, y)} \\
 &= p^2 \sum_{x=0}^{n-1} \binom{n-1}{x} p^x (1-p)^{n-1-x} \frac{1}{x+1} \frac{x+1}{\max(x - \alpha, y)} \\
 &\leq \frac{p}{n} \max_x \left(\frac{x+1}{\max(x - \alpha, y)} \right) \sum_{x=0}^{n-1} \binom{n}{x+1} p^{x+1} (1-p)^{n-(x+1)} \\
 &\leq \frac{3p}{n} (1 - (1-p)^n) < \frac{3p}{n}. \tag{13}
 \end{aligned}$$

Equation (13) follows by the following observation: $x+1 \leq 3(x-\alpha)$ for $x \geq 2$, and $x+1 \leq 2y$ for $x \leq 1$. Finally, $pE \left[\ln \frac{\max(B_{n-1} - \alpha + B_1, y)}{\max(B_{n-1} - \alpha, y)} \right] \geq 0$, which implies the lower bound of the lemma. ■

The following lemma bounds n_2 as a function of n_1 .

Lemma 34 *Let $\delta > 0$. We have $\forall^\delta S$: $n_2 = O \left(\left(\sqrt{m \ln \frac{1}{\delta}} + n_1 \right) \ln \frac{m}{\delta} \right)$.*

Theorem 32 Proof Let $y_w = \left(1 - \sqrt{\frac{3}{5}}\right) p_w m - 2$. By Lemma 9, with $\lambda = 5$, we have $\forall^{\frac{\delta}{2}} S$:

$$\forall w \in V : p_w > \frac{3 \ln \frac{4m}{\delta}}{m}, \quad \left| p_w - \frac{c_w}{m} \right| \leq \sqrt{\frac{3p_w \ln \frac{4m}{\delta}}{m}} \tag{14}$$

$$\forall w \in V : p_w > \frac{5 \ln \frac{4m}{\delta}}{m}, \quad c_w > y_w + 2 \geq (5 - \sqrt{15}) \ln \frac{4m}{\delta} > \ln \frac{4m}{\delta}. \tag{15}$$

Let $V_H = \left\{ w \in V : p_w > \frac{5 \ln \frac{4m}{\delta}}{m} \right\}$ and $V_L = V \setminus V_H$. We have

$$|L - L_{leave-one}| \leq \left| \sum_{w \in V_H} \left(p_w L_w - \frac{c_w}{m} L'_w \right) \right| + \left| \sum_{w \in V_L} \left(p_w L_w - \frac{c_w}{m} L'_w \right) \right|. \tag{16}$$

We start by bounding the first term of Equation (16). By Equation (15), we have $\forall w \in V_H, c_w > y_w + 2 > \ln \frac{4m}{\delta}$. Equation (10) implies that $q_w = \frac{c_w - \alpha}{m - \beta}$, therefore $L_w = \ln \frac{m - \beta}{c_w - \alpha} = \ln \frac{m - \beta}{\max(c_w - \alpha, y_w)}$, and $L'_w = \ln \frac{m - 1 - \beta}{c_w - 1 - \alpha} = \ln \frac{m - 1 - \beta}{\max(c_w - 1 - \alpha, y_w)}$. Let

$$Err_w^H = \frac{c_w}{m} \ln \frac{m - \beta}{\max(c_w - 1 - \alpha, y_w)} - p_w \ln \frac{m - \beta}{\max(c_w - \alpha, y_w)}.$$

We have

$$\begin{aligned}
 \left| \sum_{w \in V_H} \left(\frac{c_w}{m} L'_w - p_w L_w \right) \right| &= \left| \sum_{w \in V_H} Err_w^H + \ln \frac{m-1-\beta}{m-\beta} \sum_{w \in V_H} \frac{c_w}{m} \right| \\
 &\leq \left| \sum_{w \in V_H} Err_w^H \right| + O\left(\frac{1}{m}\right). \tag{17}
 \end{aligned}$$

We bound $\left| \sum_{w \in V_H} Err_w^H \right|$ using McDiarmid's inequality. As in Lemma 33, let $f_w(x) = \ln(\max(x, y_w))$. We have

$$E [Err_w^H] = \ln(m-\beta) E \left[\frac{c_w}{m} - p_w \right] + E \left[p_w f_w(c_w - \alpha) - \frac{c_w}{m} f_w(c_w - 1 - \alpha) \right].$$

The first expectation equals 0, the second can be bounded using Lemma 33:

$$\begin{aligned}
 \left| \sum_{w \in V_H} E [Err_w^H] \right| &\leq \sum_{w \in V_H} \left| E \left[p_w f_w(c_w - \alpha) - \frac{c_w}{m} f_w(c_w - 1 - \alpha) \right] \right| \\
 &\leq \sum_{w \in V_H} \frac{3p_w}{m} = O\left(\frac{1}{m}\right). \tag{18}
 \end{aligned}$$

In order to use McDiarmid's inequality, we bound the change of $\sum_{w \in V_H} Err_w^H$ as a function of a single change in the sample. Suppose that a word u is replaced by a word v . This results in decrease for c_u , and increase for c_v . Recalling that $y_w = \Omega(mp_w)$, the change of Err_u^H , as well as the change of Err_v^H , is bounded by $O\left(\frac{\ln m}{m}\right)$, as follows:

The change of $p_u \ln \frac{m-\beta}{\max(c_u-\alpha, y_u)}$ would be 0 if $c_u - \alpha \leq y_u$. Otherwise,

$$\begin{aligned}
 &\left| p_u \ln \frac{m-\beta}{\max(c_u-1-\alpha, y_u)} - p_u \ln \frac{m-\beta}{\max(c_u-\alpha, y_u)} \right| \\
 &\leq p_u [\ln(c_u - \alpha) - \ln(c_u - 1 - \alpha)] = p_u \ln \left(1 + \frac{1}{c_u - 1 - \alpha} \right) = O\left(\frac{p_u}{c_u}\right).
 \end{aligned}$$

Since $c_u \geq y_u = \Omega(mp_u)$, the change is bounded by $O\left(\frac{p_u}{c_u}\right) = O\left(\frac{1}{m}\right)$. The change of $\frac{c_u}{m} \ln \frac{m-\beta}{\max(c_u-1-\alpha, y_u)}$ would be $O\left(\frac{\ln m}{m}\right)$ if $c_u - 1 - \alpha \leq y_u$. Otherwise,

$$\begin{aligned}
 &\left| \frac{c_u-1}{m} \ln \frac{m-\beta}{\max(c_u-2-\alpha, y_u)} - \frac{c_u}{m} \ln \frac{m-\beta}{\max(c_u-1-\alpha, y_u)} \right| \\
 &\leq \frac{c_u-1}{m} \left| \ln \frac{m-\beta}{\max(c_u-2-\alpha, y_u)} - \ln \frac{m-\beta}{\max(c_u-1-\alpha, y_u)} \right| + \frac{1}{m} \ln \frac{m-\beta}{\max(c_u-1-\alpha, y_u)} \\
 &\leq \frac{c_u-1}{m} \ln \left(1 + \frac{1}{c_u-2-\alpha} \right) + \frac{\ln m}{m} = O\left(\frac{\ln m}{m}\right).
 \end{aligned}$$

The change of Err_v^H is bounded in a similar way.

By Equations (17) and (18), and Lemma 2, we have $\forall \frac{\delta}{16} \mathcal{S}$:

$$\begin{aligned}
 & \left| \sum_{w \in V_H} \left(\frac{c_w}{m} L'_w - p_w L_w \right) \right| \\
 & \leq \left| \sum_{w \in V_H} \text{Err}_w^H - E \left[\sum_{w \in V_H} \text{Err}_w^H \right] \right| + \left| E \left[\sum_{w \in V_H} \text{Err}_w^H \right] \right| + O\left(\frac{1}{m}\right) \\
 & \leq O\left(\frac{\ln m}{m} \sqrt{m \ln \frac{1}{\delta}} + \frac{1}{m} + \frac{1}{m}\right) = O\left(\sqrt{\frac{(\ln m)^2 \ln \frac{1}{\delta}}{m}}\right). \tag{19}
 \end{aligned}$$

Next, we bound the second term of Equation (16). By Lemma 10, we have $\forall \frac{\delta}{4} \mathcal{S}$:

$$\forall w \in V \text{ s.t. } p_w \leq \frac{3 \ln \frac{4m}{\delta}}{m}, \quad c_w \leq 6 \ln \frac{4m}{\delta}. \tag{20}$$

Let $b = 5 \ln \frac{4m}{\delta}$. By Equations (14) and (20), for any w such that $p_w \leq \frac{b}{m}$, we have

$$\frac{c_w}{m} \leq \max \left\{ p_w + \sqrt{\frac{3p_w \ln \frac{4m}{\delta}}{m}}, \frac{6 \ln \frac{4m}{\delta}}{m} \right\} \leq \frac{(5 + \sqrt{3 * 5}) \ln \frac{4m}{\delta}}{m} < \frac{2b}{m}.$$

Therefore $\forall w \in V_L$, we have $c_w < 2b$. Let $n_k^L = |V_L \cap S_k|$, $G_{k-1}^L = \frac{k}{m-k+1} n_k^L$, and $M_k^L = \sum_{w \in V_L \cap S_k} p_w$. We have

$$\begin{aligned}
 & \left| \sum_{w \in V_L} \left(\frac{c_w}{m} L'_w - p_w L_w \right) \right| \\
 & = \left| \sum_{k=1}^{2b} \frac{kn_k^L}{m} L'(k) - \sum_{k=0}^{2b-1} M_k^L L(k) \right| \\
 & \leq \left| \sum_{k=1}^{2b} \frac{kn_k^L}{m-k+1} L'(k) - \sum_{k=0}^{2b-1} M_k^L L(k) \right| + \sum_{k=1}^{2b} kn_k^L L'(k) \left(\frac{1}{m-k+1} - \frac{1}{m} \right) \\
 & = \left| \sum_{k=1}^{2b} G_{k-1}^L L'(k) - \sum_{k=0}^{2b-1} M_k^L L(k) \right| + O\left(\frac{bL_{\max}}{m}\right) \\
 & = \left| \sum_{k=0}^{2b-1} G_k^L L'(k+1) - \sum_{k=0}^{2b-1} M_k^L L(k) \right| + O\left(\frac{bL_{\max}}{m}\right) \\
 & \leq \sum_{k=0}^{2b-1} G_k^L |L'(k+1) - L(k)| + \sum_{k=0}^{2b-1} |G_k^L - M_k^L| L(k) + O\left(\frac{bL_{\max}}{m}\right). \tag{21}
 \end{aligned}$$

The first sum of Equation (21) is bounded using Equations (11) and (12), and Lemma 34:

$$\begin{aligned}
 & \sum_{k=0}^{2b-1} G_k^L |L'(k+1) - L(k)| \\
 &= \sum_{k=2}^{2b-1} G_k^L |L'(k+1) - L(k)| + G_0 |L'(1) - L(0)| + G_1 |L'(2) - L(1)|. \tag{22}
 \end{aligned}$$

The first term of Equation (22) is bounded by Equation (11):

$$\sum_{k=2}^{2b-1} G_k^L |L'(k+1) - L(k)| \leq \sum_{k=2}^{2b-1} G_k^L \cdot O\left(\frac{1}{m}\right) = O\left(\frac{1}{m}\right). \tag{23}$$

The other two terms are bounded using Lemma 34. For $n_1 > 0$, we have $\forall \frac{\delta}{16} S$, $n_2 = O\left(b\left(\sqrt{m \ln \frac{1}{\delta}} + n_1\right)\right)$. By Equation (12), we have

$$\begin{aligned}
 & G_0 |L'(1) - L(0)| + G_1 |L'(2) - L(1)| \\
 & \leq \frac{n_1}{m} \cdot O\left(\frac{1}{n_1}\right) + \frac{2n_2}{m-1} \cdot O\left(\frac{1}{n_1}\right) = O\left(b\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right). \tag{24}
 \end{aligned}$$

For $n_1 = 0$, Lemma 34 results in $n_2 = O\left(b\sqrt{m \ln \frac{1}{\delta}}\right)$, and Equation (24) transforms into

$$G_1 |L'(2) - L(1)| \leq \frac{2n_2 L_{max}}{m-1} = O\left(bL_{max}\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right). \tag{25}$$

Equations (22), (23), (24), and (25) sum up to

$$\sum_{k=0}^{2b-1} G_k^L |L'(k+1) - L(k)| = O\left(bL_{max}\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right). \tag{26}$$

The second sum of Equation (21) is bounded using Lemma 28 separately for every $k < 2b$ with accuracy $\frac{\delta}{16b}$. Since the proof of Lemma 28 also holds for G_k^L and M_k^L (instead of G_k and M_k), we have $\forall \frac{\delta}{8} S$, for every $k < 2b$, $|G_k^L - M_k^L| = O\left(b\sqrt{\frac{\ln \frac{b}{\delta}}{m}}\right)$. Therefore, together with Equations (21) and (26), we have

$$\begin{aligned}
 \left| \sum_{w \in V_L} \left(\frac{c_w}{m} L'_w - p_w L_w\right) \right| & \leq O\left(bL_{max}\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right) + \sum_{k=0}^{2b-1} L(k) O\left(b\sqrt{\frac{\ln \frac{b}{\delta}}{m}}\right) + O\left(\frac{bL_{max}}{m}\right) \\
 & = O\left(L_{max}\sqrt{\frac{b^4 \ln \frac{b}{\delta}}{m}}\right). \tag{27}
 \end{aligned}$$

The proof follows by combining Equations (16), (19), and (27). ■

5. Log-Loss A Priori

Section 4 bounds the error of the leave-one-out estimation of the log-loss. It shows that the log-loss can be effectively estimated, for a general family of learning algorithms.

Another question to be considered is the log-loss distribution itself, without the empirical estimation. That is, how large (or low) is it expected to be, and which parameters of the distribution affect it.

We denote the learning error (equivalent to the log-loss) as the KL-divergence between the true and the estimated distribution. We refer to a general family of learning algorithms, and show lower and upper bounds for the learning error.

The upper bound (Theorem 39) can be divided to three parts. The first part is the missing mass. The other two build a trade-off between a threshold (lower thresholds leads to a lower bound), and the number of words with probability exceeding this threshold (fewer words lead to a lower bound). It seems that this number of words is a necessary lower bound, as we show at Theorem 35.

Theorem 35 *Let the distribution be uniform: $\forall w \in V : p_w = \frac{1}{N}$, with $N \ll m$. Also, suppose that the learning algorithm just uses maximum-likelihood approximation, meaning $q_w = \frac{c_w}{m}$. Then, a typical learning error would be $\Omega(\frac{N}{m})$.*

The proof of Theorem 35 bases on the Pinsker inequality (Lemma 36). It first shows a lower bound for L_1 norm between the true and the expected distributions, and then transforms it to the form of the learning error.

Lemma 36 (*Pinsker Inequality*) *Given any two distributions P and Q , we have*

$$KL(P||Q) \geq \frac{1}{2}(L_1(P, Q))^2.$$

Theorem 35 Proof We first show that $L_1(P, Q)$ concentrates near $\Omega\left(\sqrt{\frac{N}{m}}\right)$. Then, we use Pinsker inequality to show lower bound⁵ of $KL(P||Q)$.

First we find a lower bound for $E[|p_w - q_w|]$. Since c_w is a binomial random variable, $\sigma^2[c_w] = mp_w(1 - p_w) = \Omega\left(\frac{m}{N}\right)$, and with some constant probability, $|c_w - mp_w| > \sigma[c_w]$. Therefore, we have

$$\begin{aligned} E[|q_w - p_w|] &= \frac{1}{m}E[|c_w - mp_w|] \\ &\geq \frac{1}{m}\sigma[c_w]P(|c_w - mp_w| > \sigma[c_w]) = \Omega\left(\frac{1}{m}\sqrt{\frac{m}{N}}\right) = \Omega\left(\frac{1}{\sqrt{mN}}\right) \end{aligned}$$

$$E\left[\sum_{w \in V} |p_w - q_w|\right] = \Omega\left(N\frac{1}{\sqrt{mN}}\right) = \Omega\left(\sqrt{\frac{N}{m}}\right).$$

5. This proof does not optimize the constants. Asymptotic analysis of logarithmic transform of binomial variables by Flajolet (1999) can be used to achieve explicit values for $KL(P||Q)$.

A single change in the sample changes $L_1(P, Q)$ by at most $\frac{2}{m}$. Using McDiarmid inequality (Lemma 2) on $L_1(P, Q)$ as a function of sample words, we have $\forall \frac{1}{2}S$:

$$\begin{aligned} L_1(P, Q) &\geq E[L_1(P, Q)] - |L_1(P, Q) - E[L_1(P, Q)]| \\ &= \Omega\left(\sqrt{\frac{N}{m}}\right) - O\left(\frac{\sqrt{m}}{m}\right) = \Omega\left(\sqrt{\frac{N}{m}}\right). \end{aligned}$$

Using Pinsker inequality (Lemma 36), we have

$$\forall \frac{1}{2}S, \quad \sum_{w \in V} p_w \ln \frac{p_w}{q_w} \geq \frac{1}{2} \left(\sum_{w \in V} |p_w - q_w| \right)^2 = \Omega\left(\frac{N}{m}\right),$$

which completes the proof. ■

Definition 37 Let $\alpha \in (0, 1)$ and $\tau \geq 1$. We define an (absolute discounting) algorithm $A_{\alpha, \tau}$, which “removes” $\frac{\alpha}{m}$ probability mass from words appearing at most τ times, and uniformly spreads it among the unseen words. We denote by $n_{1 \dots \tau} = \sum_{i=1}^{\tau} n_i$ the number of words with count between 1 and τ . The learned probability Q is defined by :

$$q_w = \begin{cases} \frac{\alpha n_{1 \dots \tau}}{m n_0} & c_w = 0 \\ \frac{c_w - \alpha}{m} & 1 \leq c_w \leq \tau \\ \frac{c_w}{m} & \tau < c_w. \end{cases}$$

The α parameter can be set to some constant, or to make the missing mass match the Good-Turing missing mass estimator, that is $\frac{\alpha n_{1 \dots \tau}}{m} = \frac{n_1}{m}$.

Definition 38 Given a distribution P , and $x \in [0, 1]$, let $F_x = \sum_{w \in V: p_w \leq x} p_w$, and $N_x = |\{w \in V : p_w > x\}|$. Clearly, for any distribution P , F_x is a monotone function of x , varying from 0 to 1, and N_x is a monotone function of x , varying from N to 0. Note that N_x is bounded by $\frac{1}{x}$.

The next theorem shows an upper bound for the learning error.

Theorem 39 For any $\delta > 0$ and $\lambda > 3$, such that $\tau < (\lambda - \sqrt{3\lambda}) \ln \frac{8m}{\delta}$, the learning error of $A_{\alpha, \tau}$ is bounded $\forall \delta S$ by

$$\begin{aligned} 0 \leq \sum_{w \in V} p_w \ln \left(\frac{p_w}{q_w} \right) &\leq M_0 \ln \left(\frac{n_0 \ln \frac{4m}{\delta}}{\alpha n_{1 \dots \tau}} \right) + \frac{\lambda \ln \frac{8m}{\delta}}{1 - \alpha} \left(\sqrt{\frac{3 \ln \frac{8}{\delta}}{m}} + M_0 \right) \\ &\quad + \frac{\alpha}{1 - \alpha} F_{\lambda \ln \frac{8m}{\delta}} + \sqrt{\frac{3 \ln \frac{8}{\delta}}{m}} + \frac{3\lambda \ln \frac{8m}{\delta}}{2(\sqrt{\lambda} - \sqrt{3})^2 m} N_{\lambda \ln \frac{8m}{\delta}}. \end{aligned}$$

The proof of Theorem 39 bases directly on Lemmas 40, 41, and 43. We can rewrite this bound roughly as

$$\sum_{w \in V} p_w \ln \left(\frac{p_w}{q_w} \right) \leq \tilde{O} \left(M_0 + \frac{\lambda}{\sqrt{m}} + \frac{N_{\lambda}}{m} \right).$$

This bound implies the characteristics of the distribution influencing the log-loss. It shows that a “good” distribution can involve many low-probability words, given that the missing mass is low. However, the learning error would increase if the dictionary included many mid-range-probability words. For example, if a typical word’s probability were $m^{-\frac{3}{4}}$, the bound would become $\tilde{O} \left(M_0 + m^{-\frac{1}{4}} \right)$.

Lemma 40 *For any $\delta > 0$, the learning error for non-appearing words can be bounded with high probability by*

$$\forall^{\delta} S, \quad \sum_{w \notin S} p_w \ln \left(\frac{p_w}{q_w} \right) \leq M_0 \ln \left(\frac{n_0 \ln \frac{m}{\delta}}{\alpha n_{1\dots\tau}} \right).$$

Proof By Lemma 13, we have $\forall^{\delta} S$, the real probability of any non-appearing word does not exceed $\frac{\ln \frac{m}{\delta}}{m}$. Therefore,

$$\begin{aligned} \sum_{w \notin S} p_w \ln \left(\frac{p_w}{q_w} \right) &= \sum_{w \notin S} p_w \ln \left(p_w \frac{m}{\alpha} \frac{n_0}{n_{1\dots\tau}} \right) \\ &\leq \sum_{w \notin S} p_w \ln \left(\frac{\ln \frac{m}{\delta}}{m} \frac{m}{\alpha} \frac{n_0}{n_{1\dots\tau}} \right) = M_0 \ln \left(\frac{n_0 \ln \frac{m}{\delta}}{\alpha n_{1\dots\tau}} \right), \end{aligned}$$

which completes the proof. ■

Lemma 41 *Let $\delta > 0, \lambda > 0$. Let $V_L = \left\{ w \in V : p_w \leq \frac{\lambda \ln \frac{2m}{\delta}}{m} \right\}$, and $V'_L = V_L \cap S$. The learning error for V'_L can be bounded with high probability by*

$$\forall^{\delta} S, \quad \sum_{w \in V'_L} p_w \ln \left(\frac{p_w}{q_w} \right) \leq \frac{\lambda \ln \frac{2m}{\delta}}{1 - \alpha} \left(\sqrt{\frac{3 \ln \frac{2}{\delta}}{m}} + M_0 \right) + \frac{\alpha}{1 - \alpha} F_{\lambda \ln \frac{2m}{\delta}}.$$

Proof We use $\ln(1 + x) \leq x$.

$$\sum_{w \in V'_L} p_w \ln \frac{p_w}{q_w} \leq \sum_{w \in V'_L} p_w \frac{p_w - q_w}{q_w}.$$

For any appearing word w , $q_w \geq \frac{1 - \alpha}{m}$. Therefore,

$$\begin{aligned}
 \sum_{w \in V'_L} p_w \frac{p_w - q_w}{q_w} &\leq \frac{m}{1 - \alpha} \sum_{w \in V'_L} p_w (p_w - q_w) \\
 &= \frac{m}{1 - \alpha} \left[\sum_{w \in V'_L} p_w \left(p_w - \frac{c_w}{m} \right) + \sum_{w \in V'_L} p_w \left(\frac{c_w}{m} - q_w \right) \right] \\
 &\leq \frac{m}{1 - \alpha} \left| \sum_{w \in V'_L} p_w \left(p_w - \frac{c_w}{m} \right) \right| + \frac{m}{1 - \alpha} \sum_{w \in V'_L} p_w \frac{\alpha}{m} \\
 &\leq \frac{m}{1 - \alpha} \frac{\lambda \ln \frac{2m}{\delta}}{m} \left| \sum_{w \in V'_L} \left(p_w - \frac{c_w}{m} \right) \right| + \frac{\alpha}{1 - \alpha} \sum_{w \in V'_L} p_w \\
 &\leq \frac{\lambda \ln \frac{2m}{\delta}}{1 - \alpha} \left| \sum_{w \in V'_L} \left(p_w - \frac{c_w}{m} \right) \right| + \frac{\alpha}{1 - \alpha} F_{\frac{\lambda \ln \frac{2m}{\delta}}{m}}. \tag{28}
 \end{aligned}$$

We apply Lemma 12 on v_L , the union of words in V_L . Let $p_{v_L} = \sum_{w \in V_L} p_w$ and $c_{v_L} = \sum_{w \in V_L} c_w$. We have $\forall^\delta S$:

$$\begin{aligned}
 \left| \sum_{w \in V'_L} \left(p_w - \frac{c_w}{m} \right) \right| &= \left| \sum_{w \in V_L} \left(p_w - \frac{c_w}{m} \right) - \sum_{w \in V_L \setminus S} \left(p_w - \frac{c_w}{m} \right) \right| \\
 &\leq \left| \sum_{w \in V_L} \left(p_w - \frac{c_w}{m} \right) \right| + \sum_{w \in V_L \setminus S} p_w \\
 &\leq \left| p_{v_L} - \frac{c_{v_L}}{m} \right| + M_0 \\
 &\leq \sqrt{\frac{3 \ln \frac{2}{\delta}}{m}} + M_0. \tag{29}
 \end{aligned}$$

The proof follows combining Equations (28) and (29). ■

Lemma 42 *Let $0 < \Delta < 1$. For any $x \in [-\Delta, \Delta]$, we have $\ln(1 + x) \geq x - \frac{x^2}{2(1 - \Delta)^2}$.*

Lemma 43 *Let $\delta > 0$, $\lambda > 3$, such that $\tau < (\lambda - \sqrt{3\lambda}) \ln \frac{4m}{\delta}$. Let the high-probability words set be $V_H = \left\{ w \in V : p_w > \frac{\lambda \ln \frac{4m}{\delta}}{m} \right\}$, and $V'_H = V_H \cap S$. The learning error for V'_H can be bounded with high probability by*

$$\forall^\delta S, \quad \sum_{w \in V'_H} p_w \ln \left(\frac{p_w}{q_w} \right) \leq \sqrt{\frac{3 \ln \frac{4}{\delta}}{m}} + \frac{3\lambda \ln \frac{4m}{\delta}}{2(\sqrt{\lambda} - \sqrt{3})^2 m} N_{\frac{\lambda \ln \frac{4m}{\delta}}{m}}.$$

Proof

$$\begin{aligned} \sum_{w \in V'_H} p_w \ln \left(\frac{p_w}{q_w} \right) &= \sum_{w \in V'_H} p_w \ln \left(\frac{p_w}{\frac{c_w}{m}} \right) + \sum_{w \in V'_H} p_w \ln \left(\frac{\frac{c_w}{m}}{q_w} \right) \\ &= \sum_{w \in V'_H} p_w \ln \left(\frac{mp_w}{c_w} \right) + \sum_{w \in V'_H, c_w \leq \tau} p_w \ln \left(\frac{c_w}{c_w - \alpha} \right). \end{aligned} \quad (30)$$

Using Lemma 9 with λ , we have $\forall^{\frac{\delta}{2}} S$:

$$\begin{aligned} \forall w \in V_H, \quad \left| p_w - \frac{c_w}{m} \right| &\leq \sqrt{\frac{3p_w \ln \frac{4m}{\delta}}{m}}, \\ \forall w \in V_H, \quad c_w &\geq (\lambda - \sqrt{3\lambda}) \ln \frac{4m}{\delta}. \end{aligned} \quad (31)$$

This means that for a reasonable choice of τ (meaning $\tau < (\lambda - \sqrt{3\lambda}) \ln \frac{4m}{\delta}$), the second term of Equation (30) is 0, and $V'_H = V_H$. Also,

$$\left| \frac{\frac{c_w}{m} - p_w}{p_w} \right| \leq \frac{1}{p_w} \sqrt{\frac{3p_w \ln \frac{4m}{\delta}}{m}} \leq \sqrt{\frac{m}{\lambda \ln \frac{2m}{\delta}} \frac{3 \ln \frac{2m}{\delta}}{m}} = \sqrt{\frac{3}{\lambda}}.$$

Therefore, we can use Lemma 42 with $\Delta = \sqrt{\frac{3}{\lambda}}$:

$$\begin{aligned} \sum_{w \in V'_H} p_w \ln \left(\frac{mp_w}{c_w} \right) &= - \sum_{w \in V_H} p_w \ln \left(1 + \frac{\frac{c_w}{m} - p_w}{p_w} \right) \\ &\leq - \sum_{w \in V_H} p_w \left[\frac{\frac{c_w}{m} - p_w}{p_w} - \frac{1}{2 \left(1 - \sqrt{\frac{3}{\lambda}} \right)^2} \left(\frac{\frac{c_w}{m} - p_w}{p_w} \right)^2 \right] \\ &= \sum_{w \in V_H} \left(p_w - \frac{c_w}{m} \right) + \frac{\lambda}{2 \left(\sqrt{\lambda} - \sqrt{3} \right)^2} \sum_{w \in V_H} \frac{\left(\frac{c_w}{m} - p_w \right)^2}{p_w}. \end{aligned} \quad (32)$$

We apply Lemma 12 on the v_H , the union of all words in V_H . Let $p_{v_H} = \sum_{w \in V_H} p_w$ and $c_{v_H} = \sum_{w \in V_H} c_w$. The bound on the first term of Equation (32) is:

$$\forall^{\frac{\delta}{2}} S, \quad \left| \sum_{w \in V_H} \left(p_w - \frac{c_w}{m} \right) \right| = \left| p_{v_H} - \frac{c_{v_H}}{m} \right| \leq \sqrt{\frac{3 \ln \frac{4}{\delta}}{m}}. \quad (33)$$

Assuming that Equation (31) holds, the second term of Equation (32) can also be bounded:

$$\sum_{w \in V_H} \frac{\left(\frac{c_w}{m} - p_w\right)^2}{p_w} \leq \sum_{w \in V_H} \frac{1}{p_w} \frac{3p_w \ln \frac{4m}{\delta}}{m} = \frac{3 \ln \frac{4m}{\delta}}{m} N_{\frac{\lambda \ln \frac{4m}{\delta}}{m}}. \quad (34)$$

The proof follows by combining Equations (30), (32), (33) and (34). ■

Acknowledgments

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, by a grant from the Israel Science Foundation and an IBM faculty award. This publication only reflects the authors' views.

We are grateful to David McAllester for his important contributions in the early stages of this research.

Appendix A. Technical Proofs

A.1 Concentration Inequalities

Lemma 6 Proof We use Stirling approximation $\Gamma(x+1) = \sqrt{2\pi x} \left(\frac{x}{e}\right)^x T_x$, where

$$T_x = \exp\left(\frac{1}{12x} + O\left(\frac{1}{x^2}\right)\right).$$

$$\begin{aligned} P(X = k) &= \binom{n}{k} p^k (1-p)^{n-k} \\ &\leq \frac{\Gamma(n+1)}{\Gamma(\mu+1)\Gamma(n-\mu+1)} \left(\frac{\mu}{n}\right)^\mu \left(\frac{n-\mu}{n}\right)^{n-\mu} \\ &= \frac{\sqrt{2\pi n}}{\sqrt{2\pi\mu}\sqrt{2\pi(n-\mu)}} \frac{n^n}{\mu^\mu (n-\mu)^{n-\mu}} \frac{\mu^\mu (n-\mu)^{n-\mu}}{n^\mu} \frac{T_n}{T_\mu T_{n-\mu}} \\ &= \frac{1}{\sqrt{2\pi\mu}} \sqrt{\frac{n}{n-\mu}} \frac{T_n}{T_\mu T_{n-\mu}} \\ &= \frac{1}{\sqrt{2\pi\mu(1-p)}} \frac{T_n}{T_\mu T_{n-\mu}}. \end{aligned}$$

Clearly, for integral values of μ , the equality is achieved at $k = \mu$. ■

Lemma 8 Proof Let $m' = \sum_{w \in V} c'_w$. Using Lemma 7 for m' with $b = c = E[m'] = m$, the probability $P(m' = m)$ achieves its minimum when $\forall w \in V, p_w = \frac{1}{N}$. Under this assumption, we have $m' \sim \text{Bin}(mN, \frac{1}{N})$. Using Lemma 6, we have

$$P(m' = m) = \frac{1}{\sqrt{2\pi m N \frac{1}{N} (1 - \frac{1}{N})}} \frac{T_{mN}}{T_m T_{mN-m}} \geq \frac{1}{3\sqrt{m}}.$$

Therefore, for any distribution $\{p_w : w \in V\}$, we have

$$P(m' = m) \geq \frac{1}{3\sqrt{m}}.$$

Obviously, $E[F'] = \sum_w E[f_w(c'_w)] = E[F]$. Also, the distribution of $\{c'_w\}$ given that $m' = m$ is identical to the distribution of $\{c_w\}$, therefore the distribution of F' given that $m' = m$ is identical to the distribution of F . We have

$$\begin{aligned} P(|F' - E[F']| > \epsilon) &= \sum_i P(m' = i) P(|F' - E[F']| > \epsilon | m' = i) \\ &\geq P(m' = m) P(|F' - E[F']| > \epsilon | m' = m) \\ &= P(m' = m) P(|F - E[F]| > \epsilon) \\ &\geq \frac{1}{3\sqrt{m}} P(|F - E[F]| > \epsilon), \end{aligned}$$

which completes the proof. ■

Lemma 44 For any $\delta > 0$, and a word $w \in V$, such that $p_w \geq \frac{3 \ln \frac{2}{\delta}}{m}$, we have

$$P\left(\left|p_w - \frac{c_w}{m}\right| > \sqrt{\frac{3p_w \ln \frac{2}{\delta}}{m}}\right) \leq \delta.$$

Proof The proof follows by applying Lemma 3, substituting $\epsilon = \sqrt{3mp_w \ln \frac{2}{\delta}}$. Note that for $3 \ln \frac{2}{\delta} \leq mp_w$ we have $\epsilon \leq mp_w$:

$$\begin{aligned} P\left(\left|p_w - \frac{c_w}{m}\right| \geq \sqrt{\frac{3p_w \ln \frac{2}{\delta}}{m}}\right) &= P(|mp_w - c_w| \geq \epsilon) \\ &\leq 2 \exp\left(-\frac{\epsilon^2}{2E[c_w] + \epsilon}\right) \\ &\leq 2 \exp\left(-\frac{3mp_w \ln \frac{2}{\delta}}{3mp_w}\right) = \delta, \end{aligned}$$

which completes the proof. ■

Lemma 9 Proof There are at most m words with probability $p_w \geq \frac{3 \ln \frac{2m}{\delta}}{m}$. The first claim follows using Lemma 44 together with union bound over all these words (with accuracy $\frac{\delta}{m}$ for each word).

Using the first claim, we derive the second. We show a lower bound for $\frac{c_w}{m}$, using $\frac{\ln \frac{2m}{\delta}}{m} < \frac{1}{\lambda} p_w$:

$$\frac{c_w}{m} \geq p_w - \sqrt{\frac{3p_w \ln \frac{2m}{\delta}}{m}} > p_w - p_w \sqrt{\frac{3}{\lambda}} = \left(1 - \sqrt{\frac{3}{\lambda}}\right) p_w.$$

The final inequality follows from simple algebra. ■

Lemma 10 Proof Let $b = 3 \ln(\frac{m}{\delta})$. Note that $\delta \in [0, 1]$ and $m > 1$ yield $b > 2$. First, suppose that there are up to m words with $p_w \leq \frac{b}{m}$. For each such word, we apply Lemma 3 on c_w , with $\varepsilon = b$. We have:

$$P\left(c_w > 6 \ln \frac{m}{\delta}\right) \leq P(c_w > mp_w + \varepsilon) \leq \exp\left(-\frac{b^2}{2mp_w + b}\right) \leq \frac{\delta}{m}.$$

Since we assume that there are up to m such words, the total mistake probability is δ .

Now we assume the general case, that is, without any assumption on the number of words. Our goal is to reduce the problem to the former conditions, that is, to create a set of size m of words with probability smaller than $\frac{b}{m}$.

We first create m empty sets v_1, \dots, v_m . Let the probability of each set v_i , p_{v_i} , be the sum of the probabilities of all the words it includes. Let the actual count of v_i , c_{v_i} , be the sum of the sample counts of all the words w it includes.

We divide all the words w between these sets in a bin-packing-approximation manner. We sort the words w in decreasing probability order. Then, we do the following loop: insert the next word w to the set v_i with the currently smallest p_{v_i} .

We claim that $p_{v_i} \leq \frac{b}{m}$ for each v_i at the end of the loop. If this inequality does not hold, then some word w made this “overflow” first. Obviously, p_w must be smaller than $\frac{b}{2m}$, otherwise it would be one of the first $\frac{2m}{b} < m$ words ordered, and would enter an empty set. If $p_w < \frac{b}{2m}$ and it made an “overflow”, then the probability of each set at the moment w was entered must exceed $\frac{b}{2m}$, since w must have entered the lightest set available. This means that the total probability of all words entered by that moment was greater than $m \frac{b}{2m} > 1$.

Applying the case of m words to the sets v_1, \dots, v_m , we have $\forall \delta$: for every v_i , $c_{v_i} \leq 2b$. Also, if the count of each set v_i does not exceed $2b$, so does the count of each word $w \in v_i$. That is,

$$P\left(\exists w : p_w \leq \frac{b}{m}, c_w > 2b\right) \leq P\left(\exists v_i : p_{v_i} \leq \frac{b}{m}, c_{v_i} > 2b\right) \leq \delta,$$

which completes the proof. ■

Lemma 11 Proof By Lemma 9 with some $\lambda > 3$ (which will be set later), we have $\forall^{\frac{\delta}{2}}S$:

$$\forall w : p_w \geq \frac{3 \ln \frac{4m}{\delta}}{m}, \quad |mp_w - c_w| \leq \sqrt{3mp_w \ln \frac{4m}{\delta}}, \quad (35)$$

$$\forall w : p_w > \frac{\lambda \ln \frac{4m}{\delta}}{m}, \quad c_w > \left(1 - \sqrt{\frac{3}{\lambda}}\right) mp_w. \quad (36)$$

By Equation (35), for any word w such that $\frac{3 \ln \frac{4m}{\delta}}{m} \leq p_w \leq \frac{\lambda \ln \frac{4m}{\delta}}{m}$, we have $c_w \leq mp_w + \sqrt{3mp_w \ln \frac{4m}{\delta}} \leq (\lambda + \sqrt{3\lambda}) \ln \frac{4m}{\delta}$. By Lemma 10, we have

$$\forall^{\frac{\delta}{2}}S, \quad \forall w \text{ s.t. } p_w \leq \frac{3 \ln \frac{4m}{\delta}}{m}, \quad c_w \leq 6 \ln \frac{4m}{\delta}.$$

It means that for any $w : mp_w \leq \lambda \ln \frac{4m}{\delta}$, we have $c_w \leq (\lambda + \sqrt{3\lambda}) \ln \frac{4m}{\delta}$. This means that for any w such that $c_w > (\lambda + \sqrt{3\lambda}) \ln \frac{4m}{\delta}$, we have $mp_w > \lambda \ln \frac{4m}{\delta}$. By Equation (36), this means $mp_w \leq \frac{1}{1 - \sqrt{\frac{3}{\lambda}}} c_w$, and by Equation (35):

$$|mp_w - c_w| \leq \sqrt{3mp_w \ln \frac{4m}{\delta}} \leq \sqrt{\frac{3c_w \ln \frac{4m}{\delta}}{1 - \sqrt{\frac{3}{\lambda}}}} = \sqrt{\frac{3c_w \sqrt{\lambda} \ln \frac{4m}{\delta}}{\sqrt{\lambda} - \sqrt{3}}}.$$

Substituting $\lambda = 12$ results in

$$\forall^{\delta}S : \quad \forall w \text{ s.t. } c_w > 18 \ln \frac{4m}{\delta}, \quad |mp_w - c_w| \leq \sqrt{6c_w \ln \frac{4m}{\delta}},$$

which completes the proof. ■

Lemma 12 Proof If $p_w \geq \frac{3 \ln \frac{2}{\delta}}{m}$, we can apply Lemma 44. We have

$$\forall^{\delta}S, \quad \left| \frac{c_w}{m} - p_w \right| \leq \sqrt{\frac{3p_w \ln \frac{2}{\delta}}{m}} \leq \sqrt{\frac{3 \ln \frac{2}{\delta}}{m}}.$$

Otherwise, we can apply Lemma 10. We have:

$$\forall^{\delta}S, \quad \left| \frac{c_w}{m} - p_w \right| \leq \max \left\{ p_w, \frac{c_w}{m} \right\} \leq \frac{6 \ln \frac{m}{\delta}}{m} \leq \sqrt{\frac{3 \ln \frac{2}{\delta}}{m}},$$

which completes the proof. ■

Lemma 13 Proof Let $b = \ln \frac{m}{\delta}$. We note that there are at most $\frac{m}{b}$ words with probability $p_w \geq \frac{b}{m}$.

$$\begin{aligned} P\left(\exists w : c_w = 0, p_w \geq \frac{b}{m}\right) &\leq \sum_{w: p_w \geq \frac{b}{m}} P(c_w = 0) \\ &= \sum_{w: p_w \geq \frac{b}{m}} (1 - p_w)^m \leq \frac{m}{b} \left(1 - \frac{b}{m}\right)^m < me^{-b} = \delta, \end{aligned}$$

which completes the proof. ■

A.2 K-Hitting Mass Estimation

Lemma 21 Proof We have $\sum_{w \in V_{k,\alpha}} p_w \leq 1$. Using Lemma 19, we bound $P(c_w = k)$ and $P(c_w = k + 1)$:

$$\begin{aligned} E[M_{k,\alpha}] &= \sum_{w \in V_{k,\alpha}} p_w P(c_w = k) = O\left(\frac{1}{\sqrt{k}}\right) \\ |E[G_{k,\alpha}] - E[M_{k,\alpha}]| &= \left| \sum_{w \in V_{k,\alpha}} \left[\frac{k+1}{m-k} P(c_w = k+1) - p_w P(c_w = k) \right] \right| \\ &= \sum_{w \in V_{k,\alpha}} p_w \frac{k+1}{m-k} P(c_w = k+1) = O\left(\frac{\sqrt{k}}{m}\right). \end{aligned} \quad (37)$$

Equation (37) follows by Lemma 20. By Lemma 18, we have $|V_{k,\alpha}| = O\left(\frac{m}{k}\right)$:

$$E[G_{k,\alpha}] = \frac{k+1}{m-k} \sum_{w \in V_{k,\alpha}} P(c_w = k+1) = O\left(\frac{k}{m} \frac{m}{k} \frac{1}{\sqrt{k}}\right) = O\left(\frac{1}{\sqrt{k}}\right),$$

which completes the proof. ■

Theorem 29 Proof The proof is done by examining four cases of k . For $k \leq 18 \ln \frac{8m}{\delta}$, we can use Lemma 28. We have

$$\forall^\delta S, \quad |\tilde{M}_k - M_k| = |G_k - M_k| = O\left(\sqrt{\frac{\ln \frac{1}{\delta}}{m}} (k + \ln \frac{m}{\delta})\right) = \tilde{O}\left(\frac{1}{\sqrt{m}}\right).$$

For $18 \ln \frac{8m}{\delta} < k \leq m^{\frac{2}{5}}$, we can use Theorem 25. We have

$$\forall^\delta S, \quad |\tilde{M}_k - M_k| = |G_k - M_k| = O\left(\sqrt{\frac{\sqrt{k} \ln \frac{m}{\delta}}{m} + \frac{k \ln \frac{m}{\delta}}{m}}\right) = \tilde{O}\left(m^{-\frac{2}{5}}\right).$$

For $m^{\frac{2}{5}} < k < \frac{m}{2}$, we can use Theorem 26. We have

$$\forall^{\delta} S, \quad |\tilde{M}_k - M_k| = |\hat{M}_k - M_k| = O\left(\frac{\sqrt{k}(\ln \frac{m}{\delta})^{\frac{3}{2}}}{m} + \frac{\sqrt{\ln \frac{m}{\delta}}}{k}\right) = \tilde{O}\left(m^{-\frac{2}{5}}\right).$$

For $k \geq \frac{m}{2}$, let $\alpha = \sqrt{6 \ln \frac{8m}{\delta}}$. By Lemma 17, we have $\forall^{\frac{\delta}{2}} S, M_k = M_{k,\alpha} \wedge \hat{M}_k = \hat{M}_{k,\alpha}$. By Lemma 18, $|V_{k,\alpha}| = O\left(\frac{m}{k}\right) = O(1)$. Let c be the bound on $|V_{k,\alpha}|$. Using Lemma 12 for each $w \in V_{k,\alpha}$ with accuracy $\frac{\delta}{2c}$, we have

$$\forall^{\frac{\delta}{2}} S, \quad \forall w \in V_{k,\alpha}, \quad \left| \frac{c_w}{m} - p_w \right| = O\left(\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right).$$

Therefore, we have $\forall^{\delta} S$:

$$|\tilde{M}_k - M_k| = |\hat{M}_{k,\alpha} - M_{k,\alpha}| \leq \sum_{w \in V_{k,\alpha}} \left| \frac{k}{m} - p_w \right| X_{w,k} = O\left(\sqrt{\frac{\ln \frac{1}{\delta}}{m}}\right) = \tilde{O}\left(\frac{1}{\sqrt{m}}\right),$$

which completes the proof. ■

Theorem 30 Proof First, we show that for any two words u and v , $Cov(X_{u,k}, X_{v,k}) = \Theta\left(\frac{k}{m^2}\right)$. Note that $\{c_v | c_u = k\} \sim Bin\left(m - k, \frac{k}{m-k}\right)$. By Lemma 6, we have:

$$\begin{aligned} P(c_u = k) = P(c_v = k) &= \frac{1}{\sqrt{2\pi k \left(1 - \frac{k}{m}\right)}} \frac{T_m}{T_k T_{m-k}}, \\ P(c_v = k | c_u = k) &= \frac{1}{\sqrt{2\pi k \left(1 - \frac{k}{m-k}\right)}} \frac{T_{m-k}}{T_k T_{m-2k}}. \end{aligned} \tag{38}$$

Using $T_x = \Theta(1)$ for $x \geq k$, we have

$$\begin{aligned} Cov(X_{u,k}, X_{v,k}) &= E[X_{u,k} X_{v,k}] - E[X_{u,k}] E[X_{v,k}] \\ &= P(c_u = k) [P(c_v = k | c_u = k) - P(c_v = k)] \\ &= \frac{1}{2\pi k \sqrt{\left(1 - \frac{k}{m}\right)}} \frac{T_m}{T_k T_{m-k}} \left[\frac{1}{\sqrt{\left(1 - \frac{k}{m-k}\right)}} \frac{T_{m-k}}{T_k T_{m-2k}} - \frac{1}{\sqrt{\left(1 - \frac{k}{m}\right)}} \frac{T_m}{T_k T_{m-k}} \right] \\ &= \Theta\left(\frac{1}{k}\right) \left[\frac{1}{\sqrt{\left(1 - \frac{k}{m-k}\right)}} \frac{T_{m-k}}{T_{m-2k}} - \frac{1}{\sqrt{\left(1 - \frac{k}{m}\right)}} \frac{T_m}{T_{m-k}} \right] \end{aligned}$$

$$\begin{aligned}
 &= \Theta\left(\frac{1}{k}\right) \left[\frac{T_{m-k}}{T_{m-2k}} \left(\frac{1}{\sqrt{1-\frac{k}{m-k}}} - \frac{1}{\sqrt{1-\frac{k}{m}}} \right) \right. \\
 &\quad \left. + \frac{1}{\sqrt{1-\frac{k}{m}}} \left(\frac{T_{m-k}}{T_{m-2k}} - \frac{T_m}{T_{m-k}} \right) \right]. \tag{39}
 \end{aligned}$$

We can bound the first term of Equation (39):

$$\begin{aligned}
 \frac{1}{\sqrt{1-\frac{k}{m-k}}} - \frac{1}{\sqrt{1-\frac{k}{m}}} &= \left(\frac{\sqrt{1-\frac{k}{m}} - \sqrt{1-\frac{k}{m-k}}}{\sqrt{1-\frac{k}{m-k}} \left(1-\frac{k}{m}\right)} \right) \left(\frac{\sqrt{1-\frac{k}{m}} + \sqrt{1-\frac{k}{m-k}}}{\sqrt{1-\frac{k}{m}} + \sqrt{1-\frac{k}{m-k}}} \right) \\
 &= \Theta\left(1 - \frac{k}{m} - 1 + \frac{k}{m-k}\right) = \Theta\left(\frac{k^2}{m^2}\right). \tag{40}
 \end{aligned}$$

Since $T_x = \exp\left(\frac{1}{12x} + O\left(\frac{1}{x^2}\right)\right) = 1 + \frac{1}{12x} + O\left(\frac{1}{x^2}\right)$ for $x \geq m - 2k$ (note that $k \ll m$), we have

$$\begin{aligned}
 \frac{T_{m-k}}{T_{m-2k}} - \frac{T_m}{T_{m-k}} &= \frac{T_{m-k}^2 - T_m T_{m-2k}}{T_{m-2k} T_{m-k}} \\
 &= \frac{1}{T_{m-2k} T_{m-k}} \left[\frac{1}{6(m-k)} - \frac{1}{12m} - \frac{1}{12(m-2k)} + O\left(\frac{1}{m^2}\right) \right] \\
 &= -\Theta\left(\frac{k^2}{m^3}\right) + O\left(\frac{1}{m^2}\right). \tag{41}
 \end{aligned}$$

Combining Equations (39), (40), and (41), we have

$$\text{Cov}(X_{u,k}, X_{v,k}) = \Theta\left(\frac{1}{k}\right) \left[\Theta\left(\frac{k^2}{m^2}\right) - \Theta\left(\frac{k^2}{m^3}\right) + O\left(\frac{1}{m^2}\right) \right] = \Theta\left(\frac{k}{m^2}\right).$$

Now we show that $\sigma^2[X_{w,k}] = \Theta\left(\frac{1}{\sqrt{k}}\right)$. By Equation (38), we have

$$\sigma^2[X_{w,k}] = P(c_w = k)(1 - P(c_w = k)) = \Theta\left(\frac{1}{\sqrt{k}}\right) \left(1 - \Theta\left(\frac{1}{\sqrt{k}}\right)\right) = \Theta\left(\frac{1}{\sqrt{k}}\right).$$

Now we find a bound for $\sigma^2[M_k]$.

$$\begin{aligned}
 \sigma^2[M_k] &= \sigma^2\left[\sum_w p_w X_{w,k}\right] \\
 &= \sum_w p_w^2 \sigma^2[X_{w,k}] + \sum_{u \neq v} p_u p_v \text{Cov}(X_{u,k}, X_{v,k}) \\
 &= \frac{m}{k} \left(\frac{k}{m}\right)^2 \Theta\left(\frac{1}{\sqrt{k}}\right) + \frac{m}{k} \left(\frac{m}{k} - 1\right) \left(\frac{k}{m}\right)^2 \Theta\left(\frac{k}{m^2}\right) \\
 &= \Theta\left(\frac{\sqrt{k}}{m}\right),
 \end{aligned}$$

which completes the proof. ■

A.3 Leave-One-Out Estimation of Log-Loss

Lemma 34 Proof Using Lemma 9, we have $\forall^{\frac{\delta}{2}}: n_2 = |U \cap S_2|$ and $n_1 = |U \cap S_1|$, where $U = \{w \in V : mp_w \leq c \ln \frac{m}{\delta}\}$, for some $c > 0$. Let $n'_2 = |U \cap S_2|$ and $n'_1 = |U \cap S_1|$. Let $b = \ln \frac{m}{\delta}$.

First, we show that $E[n'_2] = O(bE[n'_1])$.

$$\begin{aligned} E[n'_2] &= \sum_{w \in U} \binom{m}{2} p_w^2 (1 - p_w)^{m-2} \\ &= \sum_{w \in U} mp_w (1 - p_w)^{m-1} \left[\frac{m-1}{2} \frac{p_w}{1 - p_w} \right] \\ &= \sum_{w \in U} mp_w (1 - p_w)^{m-1} O(b) = O(bE[n'_1]). \end{aligned}$$

Next, we bound the deviation of n'_1 and n'_2 . A single change in the sample changes n'_1 , as well as n'_2 , by at most 1. Therefore, using Lemma 2 for n'_1 and n'_2 , we have

$$\begin{aligned} \forall^{\frac{\delta}{4}} S: \quad & n'_1 \geq E[n'_1] - O\left(\sqrt{m \ln \frac{1}{\delta}}\right), \\ \forall^{\frac{\delta}{4}} S: \quad & n'_2 \leq E[n'_2] + O\left(\sqrt{m \ln \frac{1}{\delta}}\right). \end{aligned}$$

Therefore,

$$n'_2 \leq E[n'_2] + O\left(\sqrt{m \ln \frac{1}{\delta}}\right) = O\left(bE[n'_1] + \sqrt{m \ln \frac{1}{\delta}}\right) = O\left(b\left(n'_1 + \sqrt{m \ln \frac{1}{\delta}}\right)\right),$$

which completes the proof. ■

A.4 Log-Loss A Priori

Theorem 39 Proof The KL-divergence is of course non-negative. By Lemma 40, we have

$$\forall^{\frac{\delta}{4}} S, \quad \sum_{w \notin S} p_w \ln \left(\frac{p_w}{q_w} \right) \leq M_0 \ln \left(\frac{n_0 \ln \frac{4m}{\delta}}{\alpha n_{1.. \tau}} \right). \tag{42}$$

By Lemma 41 with λ , we have $\forall^{\frac{\delta}{4}} S:$

$$\sum_{w \in \mathcal{S}: p_w \leq \frac{\lambda \ln \frac{8m}{\delta}}{m}} p_w \ln \left(\frac{p_w}{q_w} \right) \leq \frac{\lambda \ln \frac{8m}{\delta}}{1 - \alpha} \left(\sqrt{\frac{3 \ln \frac{8}{\delta}}{m}} + M_0 \right) + \frac{\alpha}{1 - \alpha} F_{\frac{\lambda \ln \frac{8m}{\delta}}{m}}. \quad (43)$$

By Lemma 43 with λ , we have $\forall \frac{\delta}{2} \mathcal{S}$:

$$\sum_{w \in \mathcal{S}: p_w > \frac{\lambda \ln \frac{8m}{\delta}}{m}} p_w \ln \left(\frac{p_w}{q_w} \right) \leq \sqrt{\frac{3 \ln \frac{8}{\delta}}{m}} + \frac{3 \lambda \ln \frac{8m}{\delta}}{2(\sqrt{\lambda} - \sqrt{3})^2 m} N_{\frac{\lambda \ln \frac{8m}{\delta}}{m}}. \quad (44)$$

The proof follows by combining Equations (42), (43), and (44). ■

Lemma 42 Proof Let $f(x) = \frac{x^2}{2(1-\Delta)^2} - x + \ln(1+x)$. Then,

$$\begin{aligned} f'(x) &= \frac{x}{(1-\Delta)^2} - 1 + \frac{1}{1+x}, \\ f''(x) &= \frac{1}{(1-\Delta)^2} - \frac{1}{(1+x)^2}. \end{aligned}$$

Clearly, $f(0) = f'(0) = 0$. Also, $f''(x) \geq 0$ for any $x \in [-\Delta, \Delta]$. Therefore, $f(x)$ is non-negative in the range above, and the lemma follows. ■

References

- D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18:155–193, 1979.
- S. F. Chen. *Building Probabilistic Models for Natural Language*. PhD thesis, Harvard University, 1996.
- S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, 1998.
- K. W. Church and W. A. Gale. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5: 19–54, 1991.
- J. R. Curran and M. Osborne. A very very large corpus doesn't always yield reliable estimates. In *Proceedings of the Sixth Conference on Natural Language Learning*, pages 126–131, 2002.
- D. P. Dubhashi and D. Ranjan. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13(2):99–124, 1998.

- P. Flajolet. Singularity analysis and asymptotics of Bernoulli sums. *Theoretical Computer Science*, 215:371–381, 1999.
- I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(16):237–264, 1953.
- I. J. Good. Turing’s anticipation of empirical Bayes in connection with the cryptanalysis of the naval Enigma. *Journal of Statistical Computation and Simulation*, 66(2):101–112, 2000.
- W. Hoeffding. On the distribution of the number of successes in independent trials. *Annals of Mathematical Statistics*, 27:713–721, 1956.
- W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.
- S. B. Holden. PAC-like upper bounds for the sample complexity of leave-one-out cross-validation. In *Proceedings of the Ninth Annual ACM Workshop on Computational Learning Theory*, pages 41–50, 1996.
- S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.
- M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, 1999.
- S. Kutin. *Algorithmic Stability and Ensemble-Based Learning*. PhD thesis, University of Chicago, 2002.
- D. McAllester and L. Ortiz. Concentration inequalities for the missing mass and for histogram rule error. *Journal of Machine Learning Research, Special Issue on Learning Theory*, 4(Oct): 895–911, 2003.
- D. McAllester and R. E. Schapire. On the convergence rate of Good-Turing estimators. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 1–6, 2000.
- D. McAllester and R. E. Schapire. Learning theory and language modeling. In *Seventeenth International Joint Conference on Artificial Intelligence*, 2001.
- C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, pages 148–188. London Math. Soc. Lectures Notes 141, Cambridge University Press, 1989.
- A. Orlitsky, N. P. Santhanam, and J. Zhang. Always Good Turing: Asymptotically optimal probability estimation. *Science*, 302(Oct):427–431, 2003.

An MDP-Based Recommender System *

Guy Shani

*Computer Science Department
Ben-Gurion University
Beer-Sheva, Israel 84105*

SHANIGU@CS.BGU.AC.IL

David Heckerman

*Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA*

HECKERMA@MICROSOFT.COM

Ronen I. Brafman

*Computer Science Department
Ben-Gurion University
Beer-Sheva, Israel 84105*

BRAFMAN@CS.BGU.AC.IL

Editor: Craig Boutilier

Abstract

Typical recommender systems adopt a static view of the recommendation process and treat it as a prediction problem. We argue that it is more appropriate to view the problem of generating recommendations as a sequential optimization problem and, consequently, that Markov decision processes (MDPs) provide a more appropriate model for recommender systems. MDPs introduce two benefits: they take into account the long-term effects of each recommendation and the expected value of each recommendation. To succeed in practice, an MDP-based recommender system must employ a strong initial model, must be solvable quickly, and should not consume too much memory. In this paper, we describe our particular MDP model, its initialization using a predictive model, the solution and update algorithm, and its actual performance on a commercial site. We also describe the particular predictive model we used which outperforms previous models. Our system is one of a small number of commercially deployed recommender systems. As far as we know, it is the first to report experimental analysis conducted on a real commercial site. These results validate the commercial value of recommender systems, and in particular, of our MDP-based approach.

Keywords: recommender systems, Markov decision processes, learning, commercial applications

1. Introduction

In many markets, consumers are faced with a wealth of products and information from which they can choose. To alleviate this problem, many web sites attempt to help users by incorporating a *recommender system* (Resnick and Varian, 1997) that provides users with a list of items and/or web-pages that are likely to interest them. Once the user makes her choice, a new list of recommended items is presented. Thus, the recommendation process is a sequential process. Moreover, in many domains, user choices are sequential in nature – for example, we buy a book by the author of a recent book we liked.

*. Parts of this paper appeared in the proceedings of UAI'02 under the title "An MDP-Based Recommender System," and the proceedings of ICAPS'03 under the title "Recommendation as a Stochastic Sequential Decision Problem."

The sequential nature of the recommendation process was noticed in the past (Zimdars et al., 2001). Taking this idea one step farther, we suggest that recommendation is not simply a sequential prediction problem, but rather, a sequential decision problem. At each point the Recommender System makes a decision: which recommendation to issue. This decision should take into account the sequential process involved and the optimization criteria suitable for the recommender system, such as the profit generated from selling an item. Thus, we suggest the use of Markov decision processes (MDP) (Puterman, 1994), a well known stochastic model of sequential decisions.

With this view in mind, a more sophisticated approach to recommender systems emerges. First, one can take into account the utility of a particular recommendation – for example, we might want to recommend a product that has a slightly lower probability of being bought, but generates higher profits. Second, we might suggest an item whose immediate reward is lower, but leads to more likely or more profitable rewards in the future.

These considerations are taken into account automatically by any good or optimal policy generated for an MDP model of the recommendation process. In particular, an optimal policy will take into account the likelihood of a recommendation to be accepted by the user, the immediate value to the site of such an acceptance, and the long-term implications of this on the user’s future choices. These considerations are taken with the appropriate balance to ensure the generation of the maximal expected reward stream.

For instance, consider a site selling electronic appliances faced with the option to suggest a video camera with a success probability of 0.5, or a VCR with a probability of 0.6. The site may choose the camera, which is less profitable, because the camera has accessories that are likely to be purchased, whereas the VCR does not. If a video-game console is another option with a smaller success probability, the large profit from the likely future event of selling game cartridges may tip the balance toward this latter choice. Similarly, when the products sold are books, by recommending a book for which there is a sequel, we may increase the likelihood that this sequel will be purchased later.

Indeed, in our implemented system, we observed less obvious instances of such sequential behavior: users who purchased novels by the well-known science fiction author, Roger Zelazny, who uses many mythological themes in his writing, often later purchase books on Greek or Hindu mythology. On the other hand, users who buy mythology books do not appear to buy Roger Zelazny novels afterwards.

The benefits of an MDP-based recommender system discussed above are offset by the fact that the model parameters are unknown. Standard reinforcement learning techniques that learn optimal behaviors will not do – they take considerable time to converge and their initial behavior is random. No commercial site will deploy a system with such behavior. Thus, we must find ways for generating good initial estimates for the MDP parameters. The approach we suggest initializes a predictive model of user behavior using data gathered on the site prior to the implementation of the recommender system. We then use the predictive model to provide initial parameters for the MDP.

Our initialization process can be performed using *any* predictive model. In this paper we suggest a particular model that outperforms previous approaches. The predictive model we describe is motivated by our sequential view of the recommendation process, but constitutes an independent contribution. The model can be thought of as an n -gram model (Chen and Goodman, 1996) or, equivalently, a (first-order) Markov chain in which states correspond to sequences of events. In this paper, we emphasize the latter interpretation due to its natural relationship with an MDP. We note that Su et al. (2000) have described the use of simple n -gram models for predicting web pages.

Their methods, however, yield poor performance on our data, probably because in our case, due to the relatively limited data set, the use of the enhancement techniques discussed below is needed.

Validating recommender system algorithms is not simple. Most recommender systems, such as dependency networks (Heckerman et al., 2000), are tested on historical data for their predictive accuracy. That is, the system is trained using historical data from sites that do not provide recommendations, and tested to see whether the recommendations conform to actual user behavior. We present the results of a similar test with our system showing it to perform better than the previous leading approach.

However, predictive accuracy is not an ideal measure, as it does not test how user behavior is influenced by the system's suggestions or what percentage of recommendations are accepted by users. To obtain this data, one must employ the system at a real site with real users, and compare the performance of this site with and without the system (or with this and other systems). The extent to which such experiments are possible is limited, as commercial site owners are unlikely to allow experiments which can degrade the performance or the "look-and-feel" of their systems. However, we were able to perform a certain set of experiments using our commercial system at the online bookstore MitoS (www.mitos.co.il) by running two models simultaneously on different users: one based on a predictive model and one based on an MDP model. We were also able, for a short period, to compare user behavior with and without recommendations. These results, which to the best of our knowledge are among the first reports of online performance in a commercial site, are reported in Section 6, providing very encouraging validation to recommender systems in general, and to our sequential optimization approach in particular.

The main contributions of this paper are: (1) A novel approach to recommender systems based on an MDP model together with appropriate initialization and solution techniques. (2) A novel predictive model that outperforms previous predictive models. (3) One of a small number of commercial applications based on MDPs. (4) The first (to the best of our knowledge) experimental analysis of a commercially deployed recommender system.

We note that the use of MDPs for recommender systems was previously suggested by Bohnenberger and Jameson (2001). They used an MDP to model the process of a consumer navigating within an airport. The state of this MDP was the consumer's position and rewards were obtained when the consumer entered a store or bought an item. Recommendations were issued on a palm-top, suggesting routes and stores to visit. However, the MDP model was hand-coded and experiments were conducted with students rather than real users.

The paper is structured as follows. In Section 2 we review the necessary background on recommender systems, MDPs, and reinforcement learning. In Section 3 we describe the predictive model we constructed whose goal is to accurately predict user behavior in an environment without recommendations. In Section 4 we present our empirical evaluation of the predictive model. In Section 5 we explain how we use this predictive model as a basis for a more sophisticated MDP-based model for the recommender system. In Section 6 we provide an empirical evaluation of the actual recommender system based on data gathered from our deployed system. We conclude the paper in Section 7 discussing our current and future work.

2. Background

In this section we provide the necessary background on recommender systems, N -gram models, and MDPs.

2.1 Recommender Systems

Early in the 1990s, when the Internet became widely used as a source of information, *information explosion* became an issue that needed addressing. Many web sites presenting a wide variety of content (such as articles, news stories, or items to purchase) discovered that users had difficulties finding the items that interested them out of the total selection. *Recommender Systems* (Resnick and Varian, 1997) help users limit their search by supplying a list of items that might interest a specific user. Different approaches were suggested for supplying meaningful recommendations to users and some were implemented in modern sites (Schafer et al., 2001). Traditional data mining techniques such as association rules were tried at the early stages of the development of recommender systems. Initially, they proved to be insufficient for the task, but more recent attempts have yielded some successful systems (Kitts et al., 2000).

Approaches originating from the field of *information retrieval (IR)* rely on the *content* of the items (such as description, category, title, author) and therefore are known as *content-based recommendations* (Mooney and Roy, 2000). These methods use some similarity score to match items based on their content. Based on this score, a list of items similar to the ones the user previously selected can be supplied. *Knowledge-based* recommender systems (Burke, 2000) go one step farther by using deeper knowledge about the user and the domain. In particular, the user is able to introduce explicit information about her preferences. Thus, for instance, the user could specify interest in Thai cuisine, and the system might suggest a restaurant serving some other south-Asian cuisine.

Another possibility is to avoid using information about the content, but rather use historical data gathered from other users in order to make a recommendation. These methods are widely known as *collaborative filtering (CF)* (Resnick et al., 1994), and we discuss them in more depth below. Finally, some systems try to create hybrid models that combine collaborative filtering and content-based recommendations (Balabanovic and Shoham, 1997; Burke, 2002).

2.2 Collaborative Filtering

The collaborative filtering approach originates in human behavior: people searching for an interesting item they know little of, such as a movie to rent at the video store, tend to rely on friends to recommend items they tried and liked. The person asking for advice is using a (small) community of friends that know her taste and can therefore make good predictions as to whether she will like a certain item. Over the net however, a larger community that can recommend items to our user is available, but the persons in this large community know little or nothing about each other. Conceptually, the goal of a collaborative filtering engine is to identify those users whose taste in items is predictive of the taste of a certain person (usually called a *neighborhood*), and use their recommendations to construct a list of items interesting for her.

To build a user's neighborhood, these methods rely on a database of past users interactions with the system. Early systems used *explicit ratings*. In such systems, users grade items (e.g., 5 stars to a great movie, 1 star to a horrible one) and then receive recommendations.¹ Later systems shifted toward *implicit ratings*. A common approach assumes that people like what they buy. A binary grading method is used when a value of 1 is given to items the user has bought and 0 to other items. Many modern recommender systems successfully implement this approach. Claypool et al. (2001) have suggested the use of other implicit grading methods through a special web browser that keeps track of user behavior such as the time spent looking at the web page, the scrolling of the page by

1. An example of such a system can be found at <http://www.movielenz.umn.edu/>.

t	X_{t-2}	X_{t-1}	X_t
1	–	–	x_1
2	–	x_1	x_2
3	x_1	x_2	x_3
4	x_2	x_3	x_4

Table 1: An auto-regressive transformation of the sequence x_1, x_2, x_3, x_4 for $k = 2$.

the user, and movements of the mouse over the page. Their evaluation, however, failed to establish a method of rating that gave results consistently better than the binary method mentioned above.

As described in Breese et al. (1998), collaborative filtering systems are either memory based or model based. Memory-based systems work directly with user data. Given the selections of a given user, a memory-based system identifies similar users and makes recommendations based on the items selected by these users. Model-based systems compress such user data into a predictive model. Examples of model-based collaborative filtering systems are Bayesian networks (Breese et al., 1998) and dependency networks (Heckerman et al., 2000). In this paper, we consider model-based systems.

2.3 The Sequential Nature of the Recommendation Process

Most recommender systems work in a sequential manner: they suggest items to the user who can then accept one of the recommendations. At the next stage a new list of recommended items is calculated and presented to the user. This sequential nature of the recommendation process, where at each stage a new list is calculated based on the user’s past ratings, will lead us naturally to our reformulation of the recommendation process as a sequential optimization process.

There is yet another sequential aspect to the recommendation process. Namely, optimal recommendations may depend not only on previous items purchased, but also on the order in which those items are purchased. Zimdars et al. (2001) recognized this possible dependency and suggested the use of an auto-regressive model (a k -order Markov chain) to represent it. They divided a sequence of transactions X_1, \dots, X_T (for example, product purchases, web-page views) into cases $(X_{t-k}, \dots, X_{t-1}, X_t)$ for $t = 1, \dots, T$ as shown in Table 1. They then built a model (in particular, a dependency network) to predict the last column given the other columns, under the assumption that the cases were exchangeable. Our model will also incorporate this sequential view.

2.4 N -gram Models

N -gram models originate in the field of language modeling. They are used to predict the next word in a sentence given the last $n - 1$ words. In the simplest form of the model, probabilities for the next word are estimated via maximum likelihood; and many methods exist for improving this simple approach including skipping, clustering, and smoothing. Skipping assumes that the probability of the next word x_i depends on words other than just the previous $n - 1$. A separate model is built using skipping and then combined with the standard n -gram model. Clustering is an approach that groups some states together for purposes of predicting next states. For example, we can group items such a basketball, football, and volleyball into a “sports ball” class. Such grouping helps to address the problem of data sparsity. Smoothing is a general name for

methods that modify the estimates of probabilities to achieve higher accuracy by adjusting zero or low probabilities upward. One type of smoothing is finite mixture modeling, which combines multiple models via a convex combination. In particular, given k component models for x_i given a prior sequence $X—p_{M_1}(x_i|X), \dots, p_{M_k}(x_i|X)—$ we can define the k -component mixture model $p(x_i|X) = \pi_1 \cdot p_{M_1}(x_i|X) + \dots + \pi_k \cdot p_{M_k}(x_i|X)$, where $\sum_{i=1}^k \pi_i = 1$ are its mixture weights. Details of these and other methods are given in Chen and Goodman (1996).

2.5 MDPs

An MDP is a model for sequential stochastic decision problems. As such, it is widely used in applications where an autonomous agent is influencing its surrounding environment through actions (for example, a navigating robot). MDPs (Bellman, 1962) have been known in the literature for quite some time, but due to some fundamental problems discussed below, few commercial applications have been implemented.

An MDP is by definition a four-tuple: $\langle S, A, Rwd, tr \rangle$, where S is a set of states, A is a set of actions, Rwd is a reward function that assigns a real value to each state/action pair, and tr is the state-transition function, which provides the probability of a transition between every pair of states given each action.

In an MDP, the decision-maker's goal is to behave so that some function of its reward stream is maximized – typically the average reward or the sum of discounted reward. An optimal solution to the MDP is such a maximizing behavior. Formally, a stationary policy for an MDP π is a mapping from states to actions, specifying which action to perform in each state. Given such an optimal policy π , at each stage of the decision process, the agent need only establish what state s it is in and execute the action $a = \pi(s)$.

Various exact and approximate algorithms exist for computing an optimal policy. Below we briefly review the algorithm known as *policy-iteration* (Howard, 1960), which we use in our implementation. A basic concept in all approaches is that of the *value function*. The value function of a policy π , denoted V^π , assigns to each state s a value which corresponds to the expected infinite-horizon discounted sum of rewards obtained when using π starting from s . This function satisfies the following recursive equation:

$$V^\pi(s) = Rwd(s, \pi(s)) + \gamma \sum_{s_j \in S} tr(s, \pi(s), s_j) V^\pi(s_j) \quad (1)$$

where $0 < \gamma < 1$ is the discount factor.² An *optimal* value function, denoted V^* , assigns to each state s its value according to an optimal policy π^* and satisfies

$$V^*(s) = \max_{a \in A} [Rwd(s, a) + \gamma \sum_{s_j \in S} tr(s, a, s_j) V^*(s_j)]. \quad (2)$$

To find a π^* and V^* using the policy-iteration algorithm, we search the space of possible policies. We start with an initial policy $\pi_0(s) = \operatorname{argmax}_{a \in A} Rwd(s, a)$. At each step we compute the value

2. We use discounting mostly for mathematical convenience. True discounting of reward would have to take into account the actual time in which each book is purchased, which does not seem worth the extra effort involved.

function based on the former policy and update the policy given the new value function:

$$V_i(s) = Rwd(s, \pi_i(s)) + \gamma \sum_{s_j \in \mathcal{S}} tr(s, \pi_i(s), s_j) V_i(s_j), \quad (3)$$

$$\pi_{i+1}(s) = \operatorname{argmax}_{a \in A} [Rwd(s, a) + \gamma \sum_{s_j \in \mathcal{S}} tr(s, a, s_j) V_i(s_j)]. \quad (4)$$

These iterations will converge to an optimal policy (Howard, 1960).

Solving MDPs is known to be a polynomial problem in the number of states (via a reduction to linear programming (Puterman, 1994)). It is usually more natural to represent the problem in terms of states variables, where each state is a possible assignment to these variables and the number of states is hence exponential in the number of state variables. This well known ‘‘curse of dimensionality’’ makes algorithms based on an explicit representation of the state-space impractical. Thus, a major research effort in the area of MDPs during the last decade has been on computing an optimal policy in a tractable manner using factored representations of the state space and other techniques (for example Boutilier et al. (2000); Koller and Parr (2000)). Unfortunately, these recent methods do not seem applicable in our domain in which the structure of the state space is quite different – that is, each state can be viewed as an assignment to a very small number of variables (three in the typical case) each with very large domains. Moreover, the values of the variables (describing items bought recently) are correlated. However, we were able to exploit the special structure of our state and action spaces using different techniques. In addition, we introduce approximations that exploit the fact that most states – that is, most item sequences – are highly unlikely to occur (a detailed explanation will follow in Section 3).

MDPs extend the simpler Markov chain (MC) model – a well known model of dynamic systems. A Markov chain is simply an MDP without actions. It contains a set of states and a stochastic transition function between states. In both models the next state does not depend on any states other than the current state.

In the context of recommender systems, if we equate actions with recommendations, then an MDP can be used to model user behavior with recommendations – as we show below – whereas an MC can be used to model user behavior without recommendations. Markov chains are also closely related to n -gram models. In a bi-gram model, the choice of the next word depends probabilistically on the previous word only. Thus, a bi-gram is simply a first-order Markov chain whose states correspond to words. An n -gram is a $n - 1$ -order Markovian model in which the next state depends on the previous $n - 1$ states. Such variants of MDP-models are well known. A non-first-order Markovian model can be converted into a first-order model by making each state include information related to the previous $n - 1$ states. More general transformation techniques that attempt to reduce the size of the state space have been investigated in the literature (for example, see Bacchus et al. (1996); Thiébaux et al. (2002)).

3. The Predictive Model

Our first step is to construct a predictive model of user purchases, that is, a model that can predict what item the user will buy next. This model does not take into account its influence on the user, as it does not model the recommendation process and its effects. Nonetheless, we shall use a Markov chain, with an appropriate formulation of the state space, as our model. In Section 4 we shall

show that our predictive model outperforms previous models, and in Section 5 we shall initialize our MDP-based recommender system using this predictive model.

3.1 The Basic Model

A Markov chain is a model of system dynamics – in our case, user “dynamics.” To use it, we need to formulate an appropriate notion of a user state and to estimate the state-transition function.

States. The states in our MC model represent the relevant information that we have about the user. This information corresponds to previous choices made by users in the form of a set of ordered sequences of selections. We ignore data such as age or gender, although it could be beneficial.³ Thus, the set of states contains all possible sequences of user selections. Of course, this formulation leads to an unmanageable state space with the usual associated problems—data sparsity and MDP solution complexity. To reduce the size of the state space, we consider only sequences of at most k items, for some relatively small value of k . We note that this approach is consistent with the intuition that the near history (for example, the current user session) often is more relevant than selections made less recently (for example, past user sessions). These sequences are represented as vectors of size k . In particular, we use $\langle x_1, \dots, x_k \rangle$ to denote the state in which the user’s last k selected items were x_1, \dots, x_k . Selection sequences with $l < k$ items are transformed into a vector in which x_1 through x_{k-l} have the value *missing*. The initial state in the Markov chain is the state in which every entry has the value *missing*.⁴ In our experiments, we used values of k ranging from 1 to 5.

The Transition Function. The transition function for our Markov chain describes the probability that a user whose k recent selections were x_1, \dots, x_k will select the item x' next, denoted $tr_{MC}(\langle x_1, x_2, \dots, x_k \rangle, \langle x_2, \dots, x_k, x' \rangle)$. Initially, this transition function is unknown to us; and we would like to estimate it based on user data. As mentioned, a maximum-likelihood estimate can be used:

$$tr_{MC}(\langle x_1, x_2, x_3 \rangle, \langle x_2, x_3, x_4 \rangle) = \frac{count(\langle x_1, x_2, x_3, x_4 \rangle)}{count(\langle x_1, x_2, x_3 \rangle)} \quad (5)$$

where $count(\langle x_1, x_2, \dots, x_k \rangle)$ is the number of times the sequence x_1, x_2, \dots, x_k was observed in the data set. This model, however, still suffers from the problem of data sparsity (for example, see Sarwar et al. (2000a)) and performs poorly in practice. In the next section, we describe several techniques for improving the estimate.

3.2 Some Improvements

We experimented with several enhancements to the maximum-likelihood n -gram model on data different from that used in our formal evaluation. The improvements described and used here are those that were found to work well.

One enhancement is a form of *skipping* (Chen and Goodman, 1996), and is based on the observation that the occurrence of the sequence x_1, x_2, x_3 lends some likelihood to the sequence x_1, x_3 . That is, if a person bought x_1, x_2, x_3 , then it is likely that someone will buy x_3 after x_1 . The particular

3. Those user attributes could be incorporated into our model by adding state variables. Attributes with large domains, such as age, can be joined into a (small) number of groups (for example, age groups) to avoid an explosion of the state space. Our similarity and clustering methods (see below) can be adapted to share training data between states with different, but related, attribute values (such as age group 25-30 and age group 30-40).

4. To accommodate systems that collect explicit rather than implicit ratings, each item x_i would be replaced by an item-rating element – for example, $x_i = \text{high}$.

skipping model that we found to work well is a simple additive model. First, the count for each state transition is initialized to the number of observed transitions in the data. Then, given a user sequence x_1, x_2, \dots, x_n , we add the fractional count $1/2^{(j-(i+3))}$ to the transition from $\langle x_i, x_{i+1}, x_{i+2} \rangle$ to $\langle x_{i+1}, x_{i+2}, x_j \rangle$, for all $i+3 < j \leq n$. This fractional count corresponds to a diminishing probability of skipping a large number of transactions in the sequence. We then normalize the counts to obtain the transition probabilities:

$$tr_{MC}(s, s') = \frac{count(s, s')}{\sum_{s'} count(s, s')} \quad (6)$$

where $count(s, s')$ is the (fractional) count associated with the transition from s to s' .

A second enhancement is a form of clustering that we have not found in the literature. Motivated by properties of our domain, the approach exploits similarity of sequences. For example, the state $\langle x, y, z \rangle$ and the state $\langle w, y, z \rangle$ are similar because some of the items appearing in the former appear in the latter as well. The essence of our approach is that the likelihood of transition from s to s' can be predicted by occurrences from t to s' , where s and t are similar. In particular, we define the similarity of states s_i and s_j to be

$$sim(s_i, s_j) = \sum_{m=1}^k \delta(s_i^m, s_j^m) \cdot (m + 1) \quad (7)$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta function and s_i^m is the m th item in state s_i . This similarity is arbitrary up to a constant. In addition, we define the *similarity count* from state s to s' to be

$$simcount(s, s') = \sum_{s_i} sim(s, s_i) \cdot tr_{MC}^{old}(s_i, s') \quad (8)$$

where $tr_{MC}^{old}(s_i, s')$ is the original transition function, with or without skipping (we shall compare the models created with and without the benefit of skipping). The new transition probability from s' to s is then given by⁵

$$tr_{MC}(s, s') = \frac{1}{2} tr_{MC}^{old}(s, s') + \frac{1}{2} \frac{simcount(s, s')}{\sum_{s''} simcount(s, s'')} \quad (9)$$

A third enhancement is the use of finite mixture modeling.⁶ Similar methods are used in n -gram models, where—for example—a trigram, a bigram, and a unigram are combined into a single model. Our mixture model is motivated by the fact that larger values of k lead to states that are more informative whereas smaller values of k lead to states on which we have more statistics. To balance these conflicting properties, we mix k models, where the i th model looks at the last i transactions. Thus, for $k = 3$, we mix three models that predict the next transaction based on the last transaction, the last two transactions, and the last three transactions. In general, we can learn mixture weights from data. We can even allow the mixture weights to depend on the given case (and informal experiments on our data suggest that such context-specificity would improve predictive accuracy). Nonetheless, for simplicity, we use $\pi_1 = \dots = \pi_k = 1/k$ in our experiments. Because our primary model is based on the k last items, the generation of the models for smaller values entails little computational overhead.

5. We examined several weighing techniques and the one described yielded the best results. The use of more complex techniques as well as attempts to learn the proper weights resulted in very minor changes.

6. Note that Equation 9 is also a simple mixture model.

4. Evaluation of the Predictive Model

Before incorporating our predictive model into an MDP-based recommender system, we evaluated the accuracy of the predictive model. Our evaluation used data corresponding to user behavior on a web site (without recommendation) and employed the evaluation metrics commonly used in the collaborative filtering literature. In Section 6 we evaluate the MDP-based approach using an experimental approach in which recommendations on an e-commerce site are manipulated by our algorithms.

4.1 Data Sets

We base our evaluations on real user transactions from the Israeli online bookstore *Mitos* (www.mitos.co.il). Two data sets were used: one containing user transactions (purchases) and one containing user browsing paths obtained from web logs. We filtered out items that were bought/visited less than 100 times and users who bought/browsed no more than one item as is commonly done when evaluating predictive models (for example, Zimdars et al. (2001)). We were left with 116 items and 10820 users in the transactions data set, and 65 items and 6678 users in the browsing data set.⁷ In our browsing data, no cookies were used by the site. If the same user visited the site with a new IP address, then we would treat her as a new user. Also, activity on the same IP address was attributed to a new user whenever there were no requests for two hours. These data sets were randomly split into a training set (90% of the users) and a test set (10% of the users).

The rationale for removing items that were rarely bought is that they cannot be reliably predicted. This is a conservative approach which implies, in practice, that a rarely visited item will not be recommended by the system, at least initially.

We evaluated predictions as follows. For every user sequence t_1, t_2, \dots, t_n in the test set, we generated the following test cases:

$$\langle t_1 \rangle, \langle t_1, t_2 \rangle, \dots, \langle t_{n-k}, t_{n-k+1}, \dots, t_{n-1} \rangle \quad (10)$$

closely following tests done by Zimdars et al. (2001). For each case, we then used our various models to determine the probability distribution for t_i given $t_{i-k}, t_{i-k+1}, \dots, t_{i-1}$ and ordered the items by this distribution. Finally, we used the t_i actually observed in conjunction with the list of recommended items to compute a score for the list.

4.2 Evaluation Metrics

We used two scores: Recommendation Score (RC) (Microsoft, 2002) and Exponential Decay Score (ED) (Breese et al., 1998) with slight modifications to fit into our sequential domain.

4.2.1 RECOMMENDATION SCORE

For this measure of accuracy, a recommendation is deemed successful if the observed item t_i is among the top m recommended items (m is varied in the experiments). The score RC is the percentage of cases in which the prediction is successful. A score of 100 means that the recommendation was successful in all cases. This score is meaningful for commerce sites that require a short list of recommendations and therefore care little about the ordering of the items in the list.

7. There are more items and users in the transaction data set since we used transactions over one year, whereas browsing data was collected only during one week.

4.2.2 EXPONENTIAL DECAY SCORE

This measure of accuracy is based on the position of the observed t_i on the recommendation list, thus evaluating not only the content of the list but also the order of items in it. The underlying assumption is that users are more likely to select a recommendation near the top of the list. In particular, it is assumed that a user will actually see the m th item in the list with probability

$$p(m) = 2^{-(m-1)/(\alpha-1)}, (m \geq 1) \quad (11)$$

where α is the half-life parameter—the index of the item in the list with probability 0.5 of being seen. The score is given by

$$100 \cdot \frac{\sum_{c \in C} p(m = \text{pos}(t_i|c))}{|C|} \quad (12)$$

where C is the set of all cases, $c = t_{i-k}, t_{i-k+1}, \dots, t_{i-1}$ is a case, and $\text{pos}(t_i|c)$ is the position of the observed item t_i in the list of recommended items for c . We used $\alpha = 5$ in our experiments in order to be consistent with the experiments of Breese et al. (1998) and Zimdars et al. (2001). The relative performance of the models was not sensitive to α .

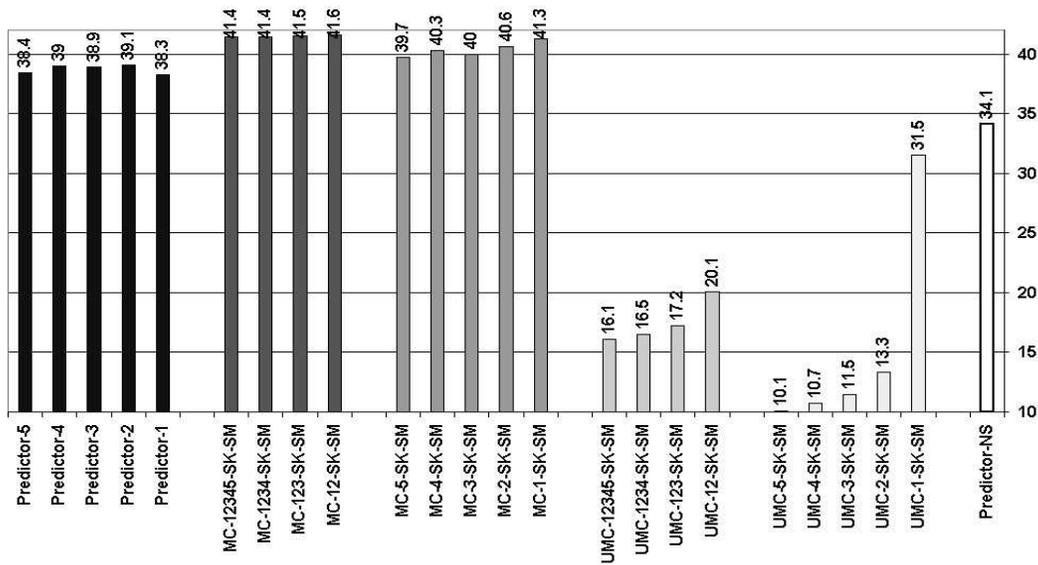
4.3 Comparison Models

4.3.1 COMMERCE SERVER 2000 PREDICTOR

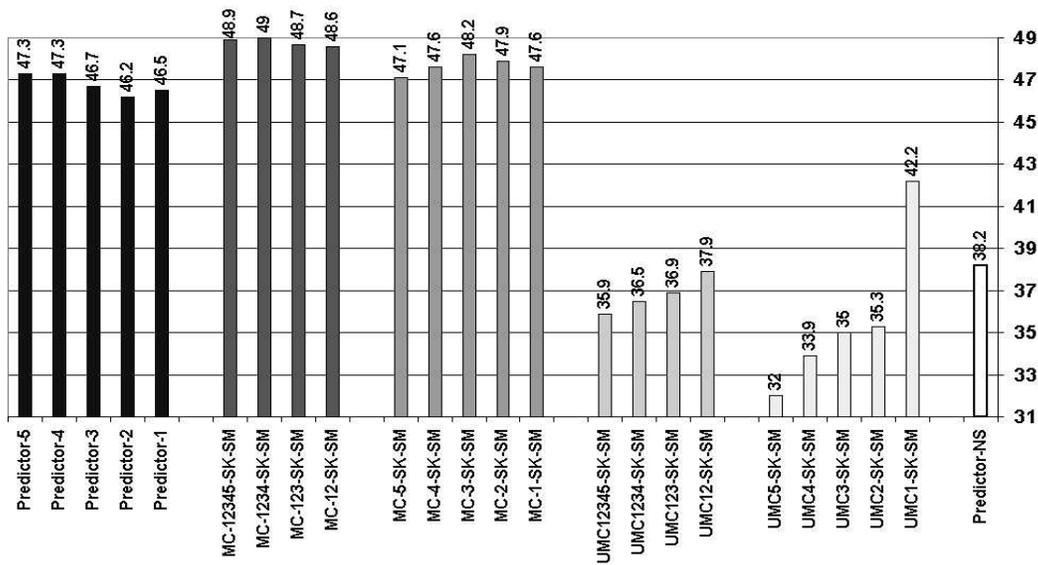
A model to which we compared our results is the *Predictor* tool developed by Microsoft as a part of Microsoft Commerce Server 2000, based on the models of Heckerman et al. (2000). This tool builds dependency-network models in which the local distributions are probabilistic decision trees. We used these models in both a non-sequential and sequential form. These two approaches are described in Heckerman et al. (2000) and Zimdars et al. (2001), respectively. In the non-sequential approach, for every item, a decision tree is built that predicts whether the item will be selected based on whether the remaining items were or were not selected. In the sequential approach, for every item, a decision tree is built that predicts whether the item will be selected next, based on the previous k items that were selected. The predictions are normalized to account for the fact that only one item can be predicted next. Zimdars et al. (2001) also use a “cache” variable, but preliminary experiments showed it to decrease predictive accuracy. Consequently, we did not use the cache variable in our formal evaluation.

These algorithms appear to be the most competitive among published work. The combined results of Breese et al. (1998) and Heckerman et al. (2000) show that (non-sequential) dependency networks are no less accurate than Bayesian-network or clustering models, and about as accurate as *Correlation*, the most accurate (but computationally expensive) memory-based method. Sarwar et al. (2000b) apply dimensionality reduction techniques to the user rating matrix, but their approach fails to be consistently more accurate than *Correlation*. Only the sequential algorithm of Zimdars et al. (2001) is more accurate than the non-sequential dependency network to our knowledge.

We built five sequential models $1 \leq k \leq 5$ for each of the data sets. We refer to the non-sequential Predictor models as Predictor-NS, and to the Predictor models built using the data expansion methods with a history of length k as Predictor- k .



(a) Transactions data set.



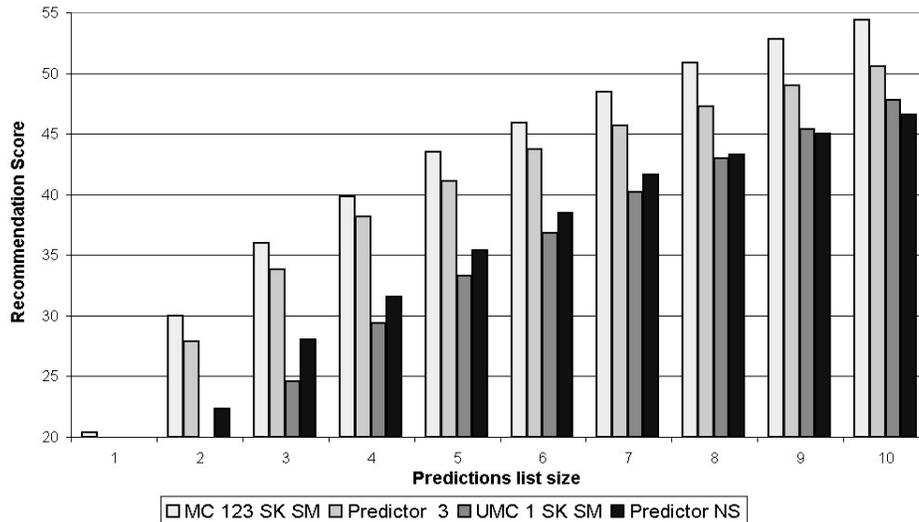
(b) Browsing data set.

Figure 1: Exponential decay score for different models.

4.3.2 UNORDERED MCs

We also evaluated a non-sequential version of our predictive model, where sequences such as $\langle x, y, z \rangle$ and $\langle y, z, x \rangle$ are mapped to the same state. If our assumption about the sequential nature of recom-

mendations is incorrect, then we should expect this model to perform better than our MC model, as it learns the probabilities using more training data for each state, gathering all the ordered data into one unordered set. Skipping, clustering, and mixture modeling were included as described in section 2. We call this model UMC (Unordered Markov chain).



(a) Transactions data set.

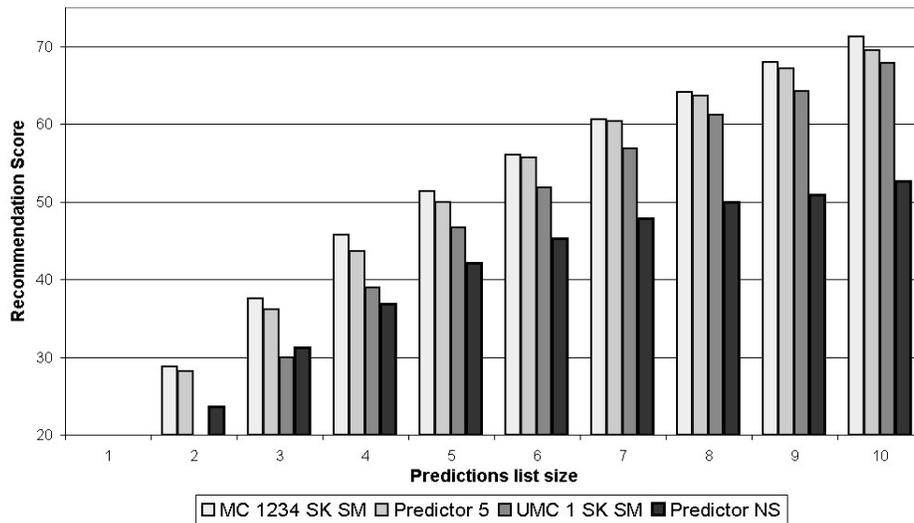
4.4 Variations of the MC Model

In order to measure how each n -gram enhancement influenced predictive accuracy, we also evaluated models that excluded some of the enhancements. In reporting our results, we refer to a model that uses skipping and similarity clustering with the terms SK and SM, respectively. In addition, we use numbers to denote which mixture components are used. Thus, for example, we use MC 123 SK to denote a Markov chain model learned with three mixture components—a bigram, trigram, and quadgram—where each component employs skipping but not clustering.

4.5 Experimental Results

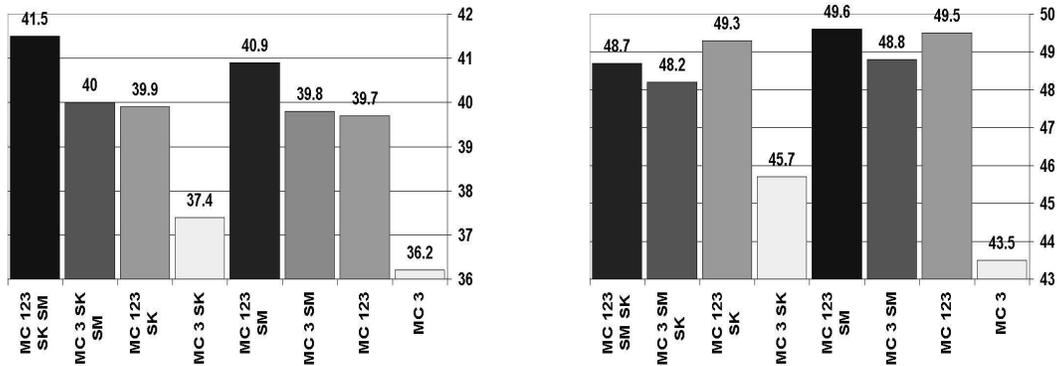
Figure 1(a) and figure 1(b) show the exponential decay score for the best models of each type (Markov chain, Unordered Markov chain, Non-Sequential Predictor model, and Sequential Predictor Model). It is important to note that *all* the MC models using skipping, clustering, and mixture modelling yielded better results than *every one of* the Predictor- k models and the non-sequential Predictor model. We see that the sequence-sensitive models are better predictors than those that ignore sequence information. Furthermore, the Markov chain predicts best for both data sets.

Figure 2(a) and Figure 2(b) show the recommendation score as a function of list length (m). Once again, sequential models are superior to non-sequential models, and the Markov chain models are superior to the Predictor models.



(b) Browsing data set.

Figure 2: Recommendation score for different models.



(a) Transactions data set.

(b) Browsing data set.

Figure 3: Exponential decay score for different Markov chain versions.

Figure 3(a) and Figure 3(b) show how different versions of the Markov chain performed under the exponential decay score in both data sets. We see that multi-component models out-perform single-component models, and that similarity clustering is beneficial. In contrast, we find that skipping is only beneficial for the transactions data set. Perhaps users tend to follow the same paths in a rather conservative manner, or site structure does not allow users to “jump ahead”. In either

case, once recommendations are available in the site (thus changing the site structure), skipping may prove beneficial.

5. An MDP-Based Recommender Model

The predictive model we described above does not attempt to capture the short and long-term effect of recommendations on the user, nor does it try to optimize its behavior by taking into account such effects. We now move to an MDP model that explicitly models the recommendation process and attempts to optimize it. The predictive model plays an important role in the construction of this model.

We assume that we are given a set of cases describing user behavior within a site that does not provide recommendations, as well as a probabilistic predictive model of a user acting without recommendations generated from this data. The set of cases is needed to support some of the approximations we make, and in particular, the lazy initialization approach we take. The predictive model provides the probability the user will purchase a particular item x given that her sequence of past purchases is x_1, \dots, x_k . We denote this value by $Pr_{pred}(x|x_1, \dots, x_k)$, where $k = 3$ in our case. It is important to stress that the approach presented here is independent of the particular technique by which the above predictive value is approximated. Naturally, in our implementation we used the predictive model developed in Section 3, but there are other ways of constructing such a model (for example, Zimdars et al. (2001); Kadie et al. (2002)).

5.1 Defining the MDP

Recall that to define an MDP, we need to provide a set of states, actions, transition function, and a reward function. We now describe each of these elements. The states of the MDP for our recommender system are k -tuples of items (for example, books, CDs), some prefix of which may contain null values corresponding to missing items. This allows us to model shorter sequences of purchases.

The actions of the MDP correspond to a recommendation of an item. One can consider multiple recommendations but, to keep our presentation simple, we start by discussing single recommendations.

Rewards in our MDP encode the utility of selling an item (or showing a web page) as defined by the site. Because the state encodes the list of items purchased, the reward depends on the last item defining the current state only. For example, the reward for state $\langle x_1, x_2, x_3 \rangle$ is the reward generated by the site from the sale of item x_3 . In this paper, we use net profit for reward.

The state following each recommendation is determined by the user's response to that recommendation. When we recommend an item x' , the user has three options:

- Accept this recommendation, thus transferring from state $\langle x_1, x_2, x_3 \rangle$ into $\langle x_2, x_3, x' \rangle$
- Select some non-recommended item x'' , thus transferring the state $\langle x_1, x_2, x_3 \rangle$ into $\langle x_2, x_3, x'' \rangle$.
- Select nothing (for example, when the user terminates the session), in which case the system remains in the same state.

Thus, the stochastic element in our model is the user's actual choice. The transition function for the MDP model:

$$tr_{MDP}^1(\langle x_1, x_2, x_3 \rangle, x', \langle x_2, x_3, x'' \rangle) \quad (13)$$

is the probability that the user will select item x'' given that item x' is recommended in state $\langle x_1, x_2, x_3 \rangle$. We write tr_{MDP}^1 to denote that only single item recommendations are used.

5.1.1 INITIALIZING tr_{MDP}

Proper initialization of the transition function is an important implementation issue in our system. Unlike traditional model-based reinforcement learning algorithms that learn the proper values for the transition function and hence an optimal policy online, our system needs to be fairly accurate when it is first deployed. A for-profit e-commerce⁸ site is unlikely to use a recommender system that generates irrelevant recommendations for a long period, while waiting for it to converge to an optimal policy. We therefore need to initialize the transition function carefully. We can do so based on any good predictive model, making the following assumptions:

- A recommendation increases the probability that a user will buy an item. This probability is proportional to the probability that the user will buy this item in the absence of recommendations. This assumption is made by most collaborative filtering models dealing with e-commerce sites.⁹ We denote the proportionality constant for recommendation r in state s by $\alpha_{s,r}$, where $\alpha_{s,r} > 1$.
- The probability that a user will buy an item that was not recommended is lower than the probability that she will buy when the system issues no recommendations at all, but still proportional to it. We denote the proportionality constant for recommendation r in state s by $\beta_{s,r}$, where $\beta_{s,r} < 1$.

To allow for a simpler representation of the equations, for a state $s = \langle x_1, \dots, x_k \rangle$ and a recommendation r let us use $s \cdot r$ to denote the state $s' = \langle x_2, \dots, x_k, r \rangle$. We use $tr_{predict}(s, s \cdot r)$ to denote the probability that the user will choose r next, given that its current state is s according to the predictive model in which recommendations are not considered, that is, $Pr_{pred}(r|s)$. Thus, with $\alpha_{s,r}$ and $\beta_{s,r}$ constant over s and r and equal to α and β , respectively, we have

$$tr_{MDP}^1(s, r, s \cdot r) = \alpha \cdot tr_{predict}(s, s \cdot r), \quad (14)$$

the probability that a user will buy r next if it was recommended;

$$tr_{MDP}^1(s, r', s \cdot r) = \beta \cdot tr_{predict}(s, s \cdot r), \quad r' \neq r, \quad (15)$$

the probability that a user will buy r if something else was recommended; and

$$tr_{MDP}^1(s, r, s) = 1 - tr_{MDP}^1(s, r, s \cdot r) - \sum_{r' \neq r} tr_{MDP}^1(s, r, s \cdot r'), \quad (16)$$

the probability that a user will not buy any new item after r was recommended. We do not see a reason to stipulate a particular relationship between α and β , although we must have

$$tr_{MDP}^1(s, r, s \cdot r) + \sum_{r' \neq r} tr_{MDP}^1(s, r', s \cdot r) < 1. \quad (17)$$

8. We use the term e-commerce, although our system, and recommender systems in general, can be used in content sites and other applications.

9. Actually CF models do not refer to the presence of recommendations, but using such systems to generate recommendations to users in commercial applications has the underlying assumption that the recommendation will increase the likelihood that a user will purchase an item.

The exact values of $\alpha_{s,r}$ and $\beta_{s,r}$ should be chosen carefully. Choosing $\alpha_{s,r}$ and $\beta_{s,r}$ to be constants over all states and recommendations (say $\alpha = 2$, $\beta = 0.5$) might cause the sum of transition probabilities in the MDP to exceed 1. The approach we took was motivated by Kitts et al. (2000), who showed that the *increase* in the probability of following a recommendation is large when one recommends items having high *lift*, defined to be $\frac{pr(x|h)}{pr(x)}$. Thus, it is not unreasonable to assume that this increase in probability is proportional to lift:

$$pr(r|s,r) - pr(r|s,r') \sim \gamma \frac{p(r|s)}{p(r)} \quad (18)$$

where $p(r)$ is the prior probability of buying r . Fixing $\alpha_{s,r}$ to be a little larger than 1 as follows:

$$\alpha_{s,r} = \frac{\gamma + p(r)}{p(r)} \quad (19)$$

where γ is a very small constant (we use $\gamma = \frac{1}{1000}$), and solving for $\beta_{s,r}$, we obtain

$$\beta_{s,r} = \frac{1 - \sum_{r'} \alpha_{s,r'} p(s \cdot r'|s)}{(n-1) p(s \cdot r|s)} + \alpha_{s,r}. \quad (20)$$

If $\beta_{s,r}$ is negative, we set it to a very small positive value and normalize the probabilities afterwards.

There are a few things to note about $tr_{MDP}^1(s, r', s \cdot r)$, the probability that a user will buy r if something else was recommended, and its representation. First, since $tr_{MDP}^1(s, r', s \cdot r) = \beta_{s,r} \cdot tr(s, s \cdot r)$, the MDP's initial transition probability does not depend on r' because our initialization is based on data that was collected without the benefit of recommendations. Of course, if one has access to data that reflects the effect of recommendations ($pr_{predict}(s \cdot r|s, r)$), one can use it to provide a more accurate initial model. Next, note that we can represent this transition function concisely using at most two values for every state-item pair: the probability that an item will be selected in a state when it is recommended (that is, $pr(s \cdot r|s, r)$) and the probability that an item will be selected when it is not recommended (that is, $pr(s \cdot r|s, r')$). Because the number of items is much smaller than the number of states, we obtain significant reduction in the space requirements of the model.

5.1.2 GENERATING MULTIPLE RECOMMENDATIONS

When moving to multiple recommendations, we make the assumption that recommendations are independent. Namely we assume that for every pair of sets of recommended items, R, R' , we have that

$$(r \in R \wedge r \in R') \vee (r \notin R \wedge r \notin R') \implies tr_{MDP}(s, R, s \cdot r) = tr_{MDP}(s, R', s \cdot r) \quad (21)$$

This assumption might prove to be false. It seems reasonable that, as the list of recommendations grows, the probability of selecting any item decreases. Another more subtle example is the case where the system “thinks” that the user is interested in an inexpensive cooking book. It can then recommend a few very expensive cooking books and one is reasonably priced (but in no way cheap) cooking book. The reasonably priced book will seem like a bargain compared to the expensive ones, thus making the user more likely to buy it.

Nevertheless, we make this assumption so as not to be forced to create a larger action space where actions are ordered combinations of recommendations. Taking the simple approach for representing the transition function we defined above, we still keep only two values for every state–item

pair:

$$tr_{MDP}(s, r \in R, s \cdot r) = tr_{MDP}^1(s, r, s \cdot r), \quad (22)$$

the probability that r will be bought if it appeared in the list of recommendations; and

$$tr_{MDP}(s, r \notin R, s \cdot r) = tr_{MDP}^1(s, r', s \cdot r) \text{ for all } r' \neq r, \quad (23)$$

the probability that r will be bought if it did not appear in the list.

As before, $tr_{MDP}(s, r \notin R, s \cdot r)$ does not depend on r , and will not depend on R in the discussion that follows. We note again, that these values are merely reasonable initial values and are adjusted by our system based on actual user behavior, as we shall discuss.

5.2 Solving the MDP

Having defined the MDP, we now consider how to solve it in order to obtain an optimal policy. Such a policy will, in effect, tell us what item to recommend given any sequence of user purchases. For the domains we studied, we found policy iteration (Howard, 1960)—with a few approximations to be described—to be a tractable solution method. In fact, on tests using real data, we found that policy iteration terminates after a few iterations. This stems from the special nature of our state space and the approximations we make, as we now explain.

Our state space enjoys a number of features that lead to fast convergence of the policy iteration algorithm:

Directionality. Transitions in our state space seem to have inherent directionality: First, a state representing a short sequence cannot follow a state representing a longer sequence. Second, the success of the sequential prediction model indicates that typically, if x is likely to follow y , y is less likely to follow x – otherwise, the sequence x, y and y, x would have similar probabilities, and we could simply use sets. Thus, loops, which in principle could occur in our MDP model because we maintain only a limited amount of history, are not very likely. Indeed, an examination of the loops in our state space graph reveals them to be small and scarce. Moreover, in the web site implementation, it is easy enough to filter out items that were already bought by the user from our list of recommendations. It is well-known that directionality can be used to reduce the running time of MDP solution algorithm (for example, Bonet and Geffner (2003)).

Insensitivity to k . We have also found that the computation of an optimal policy is not heavily sensitive to variations in k —the number of past transactions we encapsulate in a state. As k increases, so does the number of states, but the number of positive entries in our transition matrix remains similar. Note that, at most, a state can have as many successors as there are items. When k is small, the number of observed successors for a state can be large. When k grows, however, the number of successors decreases considerably. Table 2 demonstrates this relation in our implemented model.

Despite these properties of the state space, policy evaluation still requires much effort given the large state and action space we have to deal with. To alleviate this problem we resort to a number of approximations.

Ignoring Unobserved States. The vast majority of states in our models do not correspond to sequences that were observed in our training set because most combinations of items are extremely unlikely. For example, it is unlikely to find adjacent purchases of a science-fiction and a gardening book. We leverage this fact to save both space and computation time. First, we maintain transition probabilities only for states for which a transition occurred in our training data. These transitions

k	Number of states	Average number of successors
1	16,859	15.56
2	79,640	11.98
3	89,221	3.92

Table 2: The number of initialized states and the average number of state successors for different values of k .

correspond to pairs of states of the form s and $s \cdot r$. Thus, the number of transitions required per state is bounded by the number of items rather than by an amount exponential in k in the worst case. The non-zero transitions are stored explicitly, and as can be inferred from Table 2, their number is much smaller than the total number of entries in the explicit transition matrix. And while much memory is still required, in Section 6.2, we show that these requirements are not too large for modern computers to handle.

Moreover, we do not compute a policy choice for a state that was not encountered in our training data. When the value of such a state is needed for the computation of an optimal policy of some observed state, we simply use its immediate reward. That is, if the sequence $\langle x, y, z \rangle$ did not appear in the training data, we do not calculate a policy for it and assume its value to be $R(z)$ —the reward for the last item in the sequence. Note that given the skipping and clustering methods we use, the probability of making a transition from some (observed) sequence $\langle w, x, y \rangle$ to $\langle w, x, y \rangle$ is not zero even though $\langle x, y, z \rangle$ was never observed. This approximation, although risky in general MDPs, is motivated by the fact that in our initial model, for each state there is a relatively small number of items that are likely to be selected; and the probability of making a transition into an un-encountered state is very low. Moreover, the reward (that is, profit) does not change significantly across different states, so, there are no “hidden treasures” in the future that we could miss.

When a recommendation must be generated for a state that was not encountered in the past, we compute the value of the policy for this state online. This requires us to estimate the transition probabilities for a state that did not appear in our training data. We handle such new states in the same manner that we handled states for which we had sparse data in the initial predictive model – that is, using the techniques of skipping, clustering, and finite mixture of unigram, bigram, and trigrams described in Section 3.2.

Using the Independence of Recommendations. One of the basic steps in policy iteration is policy determination. At each iteration, we compute the best action for each state s – that is, the action satisfying:

$$\begin{aligned} \operatorname{argmax}_R [Rwd(s) + \gamma \sum_{s' \in S} tr(s, R, s') V_i(s')] = \\ \operatorname{argmax}_R [Rwd(s) + \gamma (\sum_{r \in R} tr_{MDP}(s, r \in R, s \cdot r) V_i(s \cdot r) + \\ \sum_{r \notin R} tr_{MDP}(s, r \notin R, s \cdot r) V_i(s \cdot r))] \end{aligned} \tag{24}$$

where $tr(s, r \in R, s \cdot r)$ and $tr(s, r \notin R, s \cdot r)$ follow the definitions above.

The above equation requires maximization over the set of possible recommendations for each state. The number of possible recommendations is n^κ , where n is the number of items and κ is the number of items we recommend each time. To handle this large action space, we make use of our

independence assumption. Recall that we assumed that the probability that a user buys a particular item depends on her current state, the item, and whether or not this item is recommended. It does not depend on the identity of the other recommended items. The following method uses this fact to quickly generate an optimal set of recommendations for each state.

Let us define $\Delta(s, r)$ – the additional value of recommending r in state s :

$$\Delta(s, r) = (tr(s, r \in R, s \cdot r) - tr(s, r \notin R, s \cdot r))V(s \cdot r). \quad (25)$$

Now define

$$R_{max\Delta}^{s,\kappa} = \{r_1, \dots, r_\kappa \mid \Delta(s, r_1) \geq \dots \geq \Delta(s, r_\kappa) \text{ and} \\ \forall r \neq r_i (i = 1, \dots, \kappa), \Delta(s, r_\kappa) \geq \Delta(s, r)\}. \quad (26)$$

$R_{max\Delta}^{s,\kappa}$ is the set of κ items that have the maximal $\Delta(s, r)$ values.

Theorem 1 $R_{max\Delta}^{s,\kappa}$ is the set that maximizes $V_{i+1}(s)$ – that is,

$$V_{i+1}(s) = Rwd(s) + \gamma(\sum_{r \in R_{max\Delta}^{s,\kappa}} tr(s, r \in R, s \cdot r)V_i(s \cdot r) + \sum_{r \notin R_{max\Delta}^{s,\kappa}} tr(s, r \notin R, s \cdot r)V_i(s \cdot r)). \quad (27)$$

Proof Let us assume that there exists some other set of κ recommendations $R \neq R_{max\Delta}^{s,\kappa}$ that maximizes $V_{i+1}(s)$. For simplicity, we shall assume that all Δ values are different. If that is not the case, then R should be a set of recommendations not equivalent to $R_{max\Delta}^{s,\kappa}$. Let r be an item in R but not in $R_{max\Delta}^{s,\kappa}$, and r' be an item in $R_{max\Delta}^{s,\kappa}$ but not in R . Let R' be the set we get when we replace r with r' in R . We need only show that $V_{i+1}(s, R) < V_{i+1}(s, R')$:

$$\begin{aligned} V_{i+1}(s, R') - V_{i+1}(s, R) &= \\ Rwd(s) + \sum_{s'} tr(s, R, s')V_i(s') - (Rwd(s) + \sum_{s'} tr(s, R', s')V_i(s')) &= \\ \sum_{r'' \in R} tr(s, r'' \in R, s \cdot r'')V_i(s \cdot r'') + \sum_{r'' \notin R} tr(s, r'' \notin R, s \cdot r'')V_i(s \cdot r'') - & \\ \sum_{r'' \in R'} tr(s, r'' \in R', s \cdot r'')V_i(s \cdot r'') - \sum_{r'' \notin R'} tr(s, r'' \notin R', s \cdot r'')V_i(s \cdot r'') &= \\ tr(s, r \in R, s \cdot r)V_i(s \cdot r) - tr(s, r' \notin R, s \cdot r')V_i(s \cdot r') - & \\ (tr(s, r' \in R', s \cdot r)V_i(s \cdot r) - tr(s, r \notin R', s \cdot r)V_i(s \cdot r)) &= \\ \Delta(s, r) - \Delta(s, r') &> 0 \end{aligned} \quad (28)$$

■

To compute $V_{i+1}(s)$ we therefore need to compute all $\Delta(s, r)$ and find $R_{max\Delta}^{s,\kappa}$, making the computation of $V_{i+1}(s)$ independent of the number of subsets (or even worse—ordered subsets) of κ items. The complexity of finding an optimal policy when recommending multiple items at each stage under our assumptions remains the same as the complexity of computing an optimal policy for single item recommendations.

By construction, our MDP optimizes site profits. In particular, the system does not recommend items that are likely to be bought whether recommended or not, but rather recommends items whose likelihood of being purchased is *increased* when they are recommended. Nonetheless, when recommendations are based solely on lift, it is possible that many recommendations will be made for which the absolute probability of a purchase (or click) is small. In this case, if recommendations are seldom followed, users might start ignoring them altogether, making the overall benefit zero. Our model does not capture such effects. One way to remedy this possible problem is to alter the reward function so as to provide a certain immediate reward for the acceptance of a recommendation. Another way to handle this problem is to recommend a book with a large MDP score only if the probability of buying it passes some threshold. We did not find it necessary to introduce these modifications in our current system.

5.3 Updating the Model Online

Once the recommender system is deployed with its initial model, we need to update the model according to actual observations. One approach is to use some form of reinforcement learning—methods that improve the model after each recommendation is made. Although such models need little administration to improve, the implementation requires many calls and computations by the recommender system online, which will lead to slower responses—an undesirable result. A simpler approach is to perform off-line updates at fixed time intervals. The site need only keep track of the recommendations and the user selections and, say, once a week use those statistics to build a new model and replace it with the old one. This is the approach we used.

In order to re-estimate the transition function the following counts are obtained from the recently collected statistics:

- $c_{in}(s, r, s \cdot r)$ —the number of times the r recommendation was accepted in state s .
- $c_{out}(s, r, s \cdot r)$ —the number of times the user took item r in state s even though it was not recommended,
- $c_{total}(s, s \cdot r)$ —the number of times a user took item r while being in state s , regardless of whether it was recommended or not.

We compute the new counts and the new approximation for the transition function at time $t + 1$ based on the counts and probabilities at time t as follows:

$$c_{in}^{t+1}(s, r, s \cdot r) = c_{in}^t(s, r, s \cdot r) + count(s, r, s \cdot r), \quad (29)$$

$$c_{total}^{t+1}(s, s \cdot r) = c_{total}^t(s, r, s \cdot r) + count(s, s \cdot r), \quad (30)$$

$$c_{out}^{t+1}(s, r, s \cdot r) = c_{out}^t(s, r, s \cdot r) + count(s, s \cdot r) - count(s, r, s \cdot r), \quad (31)$$

$$tr(s, r \in R, s \cdot r) = \frac{c_{in}^{t+1}(s, r, s \cdot r)}{c_{total}^{t+1}(s, s \cdot r)}, \quad (32)$$

$$tr(s, r \notin R, s \cdot r) = \frac{c_{out}^{t+1}(s, r, s \cdot r)}{c_{total}^{t+1}(s, s \cdot r)}. \quad (33)$$

Note that at this stage the constants $\alpha_{s,r}$ and $\beta_{s,r}$ no longer play a role—they were used only to generate the initial model. We still need to define how the counts at time $t = 0$ are initialized. We showed in section 5.1.1 how the transition function tr is initialized, and now we define:

$$c_{in}^0(s, r, s \cdot r) = \xi_s \cdot tr(s, r, s \cdot r), \quad (34)$$

$$c_{out}^0(s, r, s \cdot r) = \xi_s \cdot tr(s, r, s \cdot r), \quad (35)$$

$$c_{total}^0(s, s \cdot r) = \xi_s, \quad (36)$$

where ξ_s is proportional to the number of times the state s was observed in the training data (in our implementation we used $10 \cdot count(s)$). This initialization causes states that were observed infrequently to be updated faster than states that were observed frequently and in whose estimated transition probabilities we have more confidence.¹⁰

To ensure convergence to an optimal solution, the system must obtain accurate estimates of the transition probabilities. This, in turn, requires that for each state s and for every recommendation r , we observe the response of users to a recommendation of r in state s sufficiently many times. If at each state the system always returns the best recommendations only, then most values for $count(s, r, s \cdot r)$ would be 0, because most items will not appear among the best recommendations. Thus, the system needs to recommend non-optimal items occasionally in order to get counts for those items. This problem is widely known in computational learning as the *exploration versus exploitation tradeoff* (for some discussion of learning rate decay and exploration vs. exploitation in reinforcement learning, see, for example Kaelbling et al. (1996) and Sutton and Barto (1998)). The system balances the need to explore unobserved options in order to improve its model and the desire to exploit the data it has gathered so far in order to get rewards.

One possible solution is to select some constant ϵ , such that recommendations whose expected value is ϵ -close to optimal will be allowed—for example, by following a Boltzmann distribution:

$$Pr(choose(r_i)) = \frac{\exp \frac{V(s \cdot r_i)}{\tau}}{\sum_{j=1}^n \exp \frac{V(s \cdot r_j)}{\tau}} \quad (37)$$

with an ϵ cutoff—meaning that only items whose value is within ϵ of the optimal value will be allowed. The exact value of ϵ can be determined by the site operators. The price of such a conservative exploration policy is that we are not guaranteed convergence to an optimal policy. Another possible solution is to show the best recommendation on the top of the list, but show items less likely to be purchased as the second and third items on the list. In our implementation we use a list of three recommendations where the first one is always the optimal one, but the second and third items are selected using the Boltzmann distribution without a cutoff.

We also had to equip our system to change with frequent changes (for example, addition and removal of items). When new items are added, users will start buying them and positive counts for them will appear. At this stage, our system adds new states for these new items, and the transition function is expanded to express the transitions for these new states. Of course, prior to updating the model, the system is not able to recommend those new items (the well-known “cold start” problem (Good et al., 1999) in recommender systems). In our implementation, when the first transition to a state $s \cdot r$ is observed, its probability is initialized to 0.9 the probability of the most likely next item in state s with $\xi_s = 10$. This approach causes the new items to be recommended quite frequently.

One possible approach to handling removed items is to do nothing to our system, in which case the transition probabilities slowly decay to zero. Using this approach, however, we may still

10. This approach is similar to assigning an independent learning rate for each state and decaying it based on the amount of observed data.

insert deleted items into the list of recommended items – an undesirable feature. Consequently, in our MitoS implementation, items are programmatically removed from the model during offline updates. Another solution that we have implemented but not evaluated is to use weighted data and to exponentially decay the weights in time, thus placing more weight on more recently observed transitions.

6. Evaluation of the MDP Recommender Model

The main thesis of this work is that (1) recommendation should be viewed as a sequential optimization problem, and (2) MDPs provide an adequate model for this view. This is to be contrasted with previous systems which used predictive models for generating recommendations. In this section, we present an empirical validation of our thesis. We compare the performance of our MDP-based recommender system (denoted MDP) with the performance of a recommender system based on our predictive model (denoted MC) as well as other variants.

Our studies were performed on the online book store MitoS (www.mitos.co.il) from August, 2002 till April, 2004. During our evaluations, approximately 5000 – 6000 different users visited the *MitoS* site daily. Of those, around 900 users inserted items into their basket, thus entering our data-set.¹¹ On average, each customer inserted 1.97 items into the shopping basket. Over 15,000 items were available for purchase on the site.

Users received recommendations when adding items to the shopping cart.¹² The recommendations were based on the last k items added to the cart ordered by the time they were added. An example is shown in Figure 4 where the three book covers at the bottom are the recommended items. Every time a user was presented with a list of recommendations on either page, the system stored the recommendations that were presented and recorded whether the user purchased a recommended item. Cart deletions were rare and ignored. Once every two or three weeks, a process was run to update the model given the data that was collected over the latest time period.¹³

We compared the MDP and MC models both in terms of their value or utility to the site as well as their computational costs.

6.1 Utility Performance

Our first set of results is based on the assumption that the transition function we learn for our MDP using data collected *with* recommendations, provides the the best available model of user behavior under recommendation. Under this assumption, we can measure the effect of different recommendation policies. An important caveat is that the states in our MDP correspond to truncated (that is, last k) user sequences. Thus, the model does not exclude repeated purchases of the same item. Despite this shortcoming, we proceeded with the evaluation.

As discussed above, a predictive model can answer queries in the form $Pr(x|h)$ —the probability that item x will be purchased given user history h . Recommender systems may employ different strategies when generating recommendations using such a predictive model. Assuming that an MDP formalizes the recommendation problem well, we may use the learned MDP model to evaluate these strategies. The evaluation of the quality of different possible policies for the MDP, each corre-

11. We do not supply accurate numbers for number of users and actual profits due to the request of the site owners.

12. Users also received recommendations when looking at the description of a book, but these recommendations were based only on the user's visit to the current page and not on her cart.

13. The update process was executed by the site administrator manually and therefore the update interval varies.

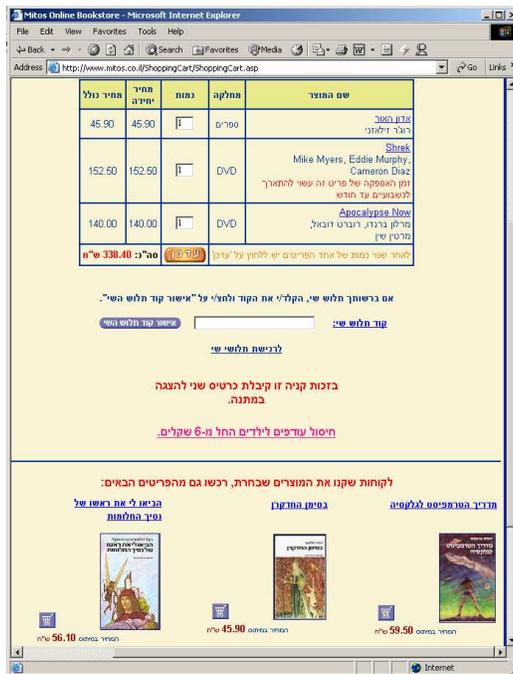


Figure 4: Recommendations in the shopping cart web page.

sponding to a popular approach to recommending, may shed light on the preferred recommendation strategy.

The MDP model was built using data gathered while the model was running in the site with incremental updates (as described above) for almost a year. We compared four policies, where the first policy uses information about the effect of recommendations, and the remaining policies are based on the predictive model solely:

- Optimal – recommends items based on optimal policy for the MDP.
- Greedy – recommends items that maximize $Pr(x|h) \cdot R(x)$ (where $Pr(x|h)$ is the probability of buying item x given user history h , and $R(x)$ is the value of x to the site – for example, net profit).
- Most likely – recommends items that maximize $Pr(x|h)$.
- Lift – recommends items that maximize $\frac{Pr(x|h)}{Pr(x)}$, where $Pr(x)$ is the prior probability of buying item x .

To evaluate the different policies we ran a simulation of the interaction of a user with the system. During the simulation the system generated a list of recommended items R , from which the simulated user selected the next item, using the distribution $tr(s, R, s \cdot x)$ —the probability that the next selected item is x given the current state s and the recommendation list R , simulating the purchase of x by the user. The length of user session was taken from the learned distribution of user session length in the actual site. We ran the simulation for 10,000 iterations for each policy, and calculated the average accumulated reward for user session.

Policy	Value
Optimal	118.5
Greedy	116.1
Most Likely	117.0
Lift	112.8

Table 3: Performance of different policies.

The results are presented in Table 3. The calculated value for each policy is the sum of discounted profit in (New Israeli Shekels) averaged over all states. We used a weighted average, where the weight of each state was the probability of observing it. Obviously, an optimal policy results in the highest value. However, the differences are small, and it appears that one can use the predictive model alone with very good results.

Next, we performed an experiment to compare the performance of the MDP-based system with that of the MC-based system. In this experiment, each user entering the site was assigned a randomly generated cart-id. Based on the last bit of this cart-id, the user was provided with recommendations by the MDP or MC. Reported mean profits were calculated for each user session (a single visit to the site). Data gathered in both cases was used to update both models.¹⁴

The deployed system was built using three mixture components, with history length ranging from one to three for both the MDP model and the MC model. Recommendations from the different mixture components were combined using an equal (0.33) weight. We used the policy-iteration procedure and approximations described in Section 5 to compute an optimal policy for the MDP. Our model encoded approximately 25,000 states in the two top mixture components ($k = 2, k = 3$). The reported results were gathered after the model was running in the site with incremental updates (as described above) for almost a year.

During the testing period, 50.7% of the users who made at least one purchase were shown MDP-based recommendations and the other 49.3% of these users were shown MC-based recommendations. For each user, we computed the average site profit per session for that user, leaving out of consideration the first purchase made in each session. The first item was excluded as it was bought without the benefit of recommendations, and is therefore irrelevant to the comparison between the recommender systems.¹⁵

The average site profit generated by the users was 28% higher for the MDP group.¹⁶ We used a permutation test (see, for example, Yeh (2000)) to see how likely it would be for a difference this large to emerge if there were in fact no systematic difference in the effectiveness of the two recommendation methods.¹⁷ We randomly generated 10000 permutations of the assignments of

14. We update the MC model by recording the transition without considering the recommendation used.

15. This is not entirely accurate as the site also provides recommendations for items in the book description page. We do not present here any experimental results for those recommendations and do not model their effect on the user, but we note that a user that received MDP recommendations in the cart page, got MDP recommendations in the book description page; users who got MC recommendations in the basket got MC recommendations in the description page as well.

16. We are not at liberty to provide accurate numbers.

17. We used a permutation test to establish the validity of our results, as this test is non-parametric, and does not require any prior assumptions about the distribution of the data, and is quite robust to noise in the data. We used the one-tailed version of the test as the directional hypothesis that the MDP recommender is better than the MC recommender has been theoretically motivated above.

session profits to users, for each permutation computing the ratio of average session profits between the MDP and the MC groups. With only 8% of these random assignments was the ratio as large as (or larger than) 1.282. Therefore, the better performance of the MDP recommender is statistically significant with $p = 0.08$ by a one-tailed permutation test.

There are two possible sources for the observed improvement—the MDP may be generating more sales or sales of more expensive items. In our experiment, the average number of items bought per user session was 6.8% in favor of the MDP-based recommender ($p = 0.15$), whereas the average price of items was 4% higher in favor of the MDP-based recommender ($p = 0.04$). Thus, both effects may have played a role.

In our second and last experiment, we compared site performance with and without a recommender system. Ideally, we would have liked to assign users randomly to an experience with and without recommendations. This option was ruled-out by the site owner because it would have led to a non-uniform user experience. Fortunately, the site owner was willing to remove the recommender system from the site for one week. Thus, we were able to compare average profits per user session during two consecutive weeks – one with recommendations and one without recommendations.¹⁸ We found that, when the recommender system was not in use, average site profit dropped 17% ($p = 0.0$). Although, we cannot rule out the possibility that this difference is due to other factors (for example, seasonal effects or special events), these result are quite encouraging.

Overall, our experiments support the claims concerning the added value of using recommendations in commercial web sites and the validity of the MDP-based model for recommender systems.

6.2 Computational Analysis

In this section, we compare computational costs of the MDP-based and the Predictor recommender system.

Our comparison uses the transaction data set and corresponding models described in Section 4. In addition to using the full data set, we measured costs associated with smaller versions of the data in which transactions among only the the top N items were considered, in order to demonstrate the effect of the size of the data-set on performance.

	$N = 15231$	$N = 2661$	$N = 1142$	$N = 354$	$N = 86$
MDP	112	63	58	41	16
Predictor-NS	3504	631	177	80	25

Table 4: Required time (seconds) for model building.

First, let us consider the time it takes to make a recommendation. Recommendation time is typically the most critical of computational costs. If recommendation latency is noticeable, no reasonable site administrator will use the recommender system. Table 5 shows the number of recommendations generated per second by the recommender system. The results show that the MDP model is faster. This result is due to the fact that, with the MDP model, we do almost no computations online. While predicting, the model simply finds the proper state and returns the state’s pre-calculated list of recommendations.

18. We display recommendations between 3/27/2003 and 4/3/2003, and without recommendations from 3/19/2003 to 3/26/2003.

	$N = 15231$	$N = 2661$	$N = 1142$	$N = 354$	$N = 86$
MDP	250	277	322	384	1030
Predictor-NS	23	74	175	322	1000

Table 5: Recommendations per second.

The price paid for faster recommendation is a larger memory footprint. Table 6 shows the amount of memory needed to build and store a model in megabytes. The MDP model requires more memory to store than the Predictor model, due to the structured representation of the Predictor model using a collection of decision trees.

Finally, we consider the time needed to build a new model. This computational cost is perhaps the least important parameter when selecting a recommender system, as model building is an off-line task executed at long time intervals (say once a week at most) on a machine that does not affect the performance of the site. That being said, as we see in Table 4, the MDP model has the smallest build times.

	$N = 15231$	$N = 2661$	$N = 1142$	$N = 354$	$N = 86$
MDP	138	74	55.7	33.3	11.4
Predictor-NS	50.1	26	25	22.3	18

Table 6: Required memory (megabytes) for building a model and generating recommendations.

Overall the MDP-based model is quite competitive with the Predictor model. It provides the fastest recommendations at the price of more memory use, and builds models more quickly.

7. Discussion

This paper describes a new model for recommender systems based on an MDP. Our work presents one of a few examples of commercial systems that use MDPs, and one of the first reports of the performance of commercially deployed recommender system. Our experimental results validate both the utility of recommender systems and the utility of the MDP-based approach to recommender systems.

To provide the kind of performance required by an online commercial site, we used various approximations and, in particular, made heavy use of the special properties of our state space and its sequential origin. Whereas the applicability of these techniques beyond recommender systems is not clear, it represents an interesting case study of a successful real system. Moreover, the sequential nature of our system stems from the fact that we need to maintain history of past purchases in order to obtain a Markovian state space. The need to record facts about the past in the current state arises in various domains, and has been discussed in a number of papers on handling non-first-order Markov reward functions (see, for example, Bacchus et al. (1996) or Thiébaux et al. (2002)).

Another interesting technique is our use of off-line data to initialize a model that can provide adequate initial performance.

In the future, we hope to improve our transition function on those states that are seldom encountered using generalization techniques, such as skipping and clustering, that are similar to the ones

we employed in the predictive Markov chain model. Other potential improvements are the use of a partially observable MDP to model the user. As a model, this is more appropriate than an MDP, as it allows us to explicitly model our uncertainty about the true state of the user (Boutilier, 2002).

In fact, our current model can be viewed as approximating a particular POMDP by using a finite – rather than an unbounded – window of past history to define the current state. Of course, the computational and representational overhead of POMDPs are significant, and appropriate techniques for overcoming these problems must be developed.

Weaknesses of our predictive (Markov chain) model include the use of *ad hoc* weighting functions for skipping and similarity functions and the use of fixed mixture weights. Although the recommendations that result from our current model are (empirically) useful for ranking items, we have noticed that the model probability distributions are not calibrated. Learning the weighting functions and mixture weights from data should improve calibration. In addition, in informal experiments, we have seen evidence that learning case-dependent mixture weights should improve predictive accuracy.

Our predictive model should also make use of relations between items that can be explicitly specified. For example, most sites that sell items have a large catalogue with hierarchical structure such as categories or subjects, a carefully constructed web structure, and item properties such as author name. Finally, our models should incorporate information about users such as age and gender.

Acknowledgments

We would like to thank the Israeli online bookstore MitoS for allowing the wide range of experiments reported in this paper, for their willingness to try new ideas and for the great effort they have put into implementing our system.

We would also like to thank the thorough reviewers of this paper that suggested many useful improvements, and helped us properly present our results in a clear manner.

Ronen Brafman is partially supported by the Paul Ivanier Center for Robotics and Production Management.

References

- F. Bacchus, C. Boutilier, and A. J. Grove. Rewarding behaviors. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, Vol. 2*, pages 1160–1167, Portland, OR, 1996.
- M. Balabanovic and Y. Shoham. Combining content-based and collaborative recommendation. *Communications of the ACM*, 40(3):62–72, March 1997.
- R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1962.
- T. Bohnenberger and A. Jameson. When policies are better than plans: decision-theoretic planning of recommendation sequences. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 21–24, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-325-1.

- B. Bonet and H. Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback. In G. Gottlob, editor, *18th International Joint Conf. on Artificial Intelligence*, pages 1233–1238, Acapulco, Mexico, 2003. Morgan Kaufmann Publishers Inc.
- C. Boutilier. A POMDP formulation of preference elicitation problems. In *Eighteenth national conference on Artificial intelligence*, pages 239–246, Edmonton, Alberta, Canada, 2002. American Association for Artificial Intelligence.
- C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000.
- J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pages 43–52, San Francisco, California, 1998. Morgan Kaufmann Publishers Inc.
- R. Burke. *Knowledge-Based Recommender Systems*, volume 69 of *Encyclopedia of Library and Information Systems*, supplement 32. A. Kent, New York, 2000.
- R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
- S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco, 1996. Morgan Kaufmann Publishers Inc.
- M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 33–40, Santa Fe, New Mexico, United States, 2001. ACM Press.
- N. Good, J. Ben Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 439–446, Orlando, Florida, United States, 1999. American Association for Artificial Intelligence.
- D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. M. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.
- R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.
- C. M. Kadie, C. Meek, and D. Heckerman. CFW: A collaborative filtering system using posteriors over weights of evidence. In *18th Conference on Uncertainty in AI (UAI'02)*, pages 242–250, 2002.
- L. P. Kaelbling, M. L. Littman, and A. P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

- B. Kitts, D. Freed, and M. Vrieze. Cross-sell: a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 437–446, Boston, Massachusetts, United States, 2000. ACM Press.
- D. Koller and R. Parr. Policy iteration for factored mdps. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 326–334, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
- Microsoft. Recommendation score. *Microsoft Commerce Server 2002 Documentation*, 2002.
- R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, San Antonio, Texas, United States, 2000. ACM Press.
- M. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.
- P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.
- P. Resnick and H. R. Varian. Recommender systems. *Special issue of Communications of the ACM*, pages 56–58, March 1997.
- B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems: a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, August 2000a.
- B. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167, New York, NY, USA, 2000b. ACM Press. ISBN 1-58113-272-7.
- J. B. Schafer, J. A. Konstan, and J. Riedle. E-commerce recommendation applications. *Data Mining Knowledge Discovery*, 5(1-2):115–153, 2001.
- Z. Su, Q. Yang, and H. J. Zhang. A prediction system for multimedia pre-fetching in internet. In *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia*, pages 3–11, Marina del Rey, California, United States, 2000. ACM Press.
- R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- S. Thiébaux, F. Kabanza, and J. Slaney. Anytime state-based solution methods for decision processes with non-Markovian rewards. In *18th Conference on Uncertainty in AI (UAI'02)*, pages 501–510, Edmonton, Canada, July 2002. Morgan Kaufmann.
- A. Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 17th conference on Computational linguistics*, pages 947–953, Saarbrcken, Germany, 2000. Association for Computational Linguistics.

- A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 580–588, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Universal Algorithms for Learning Theory

Part I : Piecewise Constant Functions

Peter Binev

BINEV@MATH.SC.EDU

*Industrial Mathematics Institute
Department of Mathematics
University of South Carolina
Columbia, SC 29208, USA*

Albert Cohen

COHEN@ANN.JUSSIEU.FR

*Laboratoire Jacques-Louis Lions
Université Pierre et Marie Curie
175, rue du Chevaleret
75013 Paris, France*

Wolfgang Dahmen

DAHMEN@IGPM.RWTH-AACHEN.DE

*Institut für Geometrie und Praktische Mathematik
RWTH Aachen
Templergraben 55
D-52056 Aachen, Germany*

Ronald DeVore

DEVORE@MATH.SC.EDU

Vladimir Temlyakov

TEMLYAK@MATH.SC.EDU

*Industrial Mathematics Institute
Department of Mathematics
University of South Carolina
Columbia, SC 29208, USA*

Editor: Peter Bartlett

Abstract

This paper is concerned with the construction and analysis of a universal estimator for the regression problem in supervised learning. Universal means that the estimator does not depend on any a priori assumptions about the regression function to be estimated. The universal estimator studied in this paper consists of a least-square fitting procedure using piecewise constant functions on a partition which depends adaptively on the data. The partition is generated by a splitting procedure which differs from those used in CART algorithms. It is proven that this estimator performs at the optimal convergence rate for a wide class of priors on the regression function. Namely, as will be made precise in the text, if the regression function is in any one of a certain class of approximation spaces (or smoothness spaces of order not exceeding one – a limitation resulting because the estimator uses piecewise constants) measured relative to the marginal measure, then the estimator converges to the regression function (in the least squares sense) with an optimal rate of convergence in terms of the number of samples. The estimator is also numerically feasible and can be implemented on-line.

Keywords: distribution-free learning theory, nonparametric regression, universal algorithms, adaptive approximation, on-line algorithms

1. Introduction

This paper addresses the problem of using empirical samples to derive probabilistic or expectation error estimates for the regression function of some unknown probability measure ρ on a product space $Z := X \times Y$. It will be assumed here that X is a bounded domain of \mathbb{R}^d and $Y = \mathbb{R}$. Given the data $\mathbf{z} = \{z_1, \dots, z_m\} \subset Z$ of m independent random observations $z_i = (x_i, y_i)$, $i = 1, \dots, m$, identically distributed according to ρ , we are interested in estimating the *regression function* $f_\rho(x)$ defined as the conditional expectation of the random variable y at x :

$$f_\rho(x) := \int_Y y d\rho(y|x)$$

with $\rho(y|x)$ the conditional probability measure on Y with respect to x . In this paper, it is assumed that this probability measure is supported on an interval $[-M, M]$:

$$|y| \leq M,$$

almost surely. It follows in particular that $|f_\rho| \leq M$ almost everywhere with respect to ρ_X .

We denote by ρ_X the marginal probability measure on X defined by

$$\rho_X(S) := \rho(S \times Y).$$

We shall assume that ρ_X is a Borel measure on X . We have

$$d\rho(x, y) = d\rho(y|x) d\rho_X(x).$$

It is easy to check that f_ρ is the minimizer of the risk functional

$$\mathcal{E}(f) := \int_Z (y - f(x))^2 d\rho, \tag{1}$$

over $f \in L_2(X, \rho_X)$ where this space consists of all functions from X to Y which are square integrable with respect to ρ_X . In fact one has

$$\mathcal{E}(f) = \mathcal{E}(f_\rho) + \|f - f_\rho\|^2,$$

where

$$\|\cdot\| := \|\cdot\|_{L_2(X, \rho_X)}. \tag{2}$$

Our objective is therefore to find an *estimator* $f_{\mathbf{z}}$ for f_ρ based on \mathbf{z} such that the quantity $\|f_{\mathbf{z}} - f_\rho\|$ is small.

A common approach to this problem is to choose an hypothesis (or *model*) class \mathcal{H} and then to define $f_{\mathbf{z}}$, in analogy to (1), as the minimizer of the empirical risk

$$f_{\mathbf{z}} = f_{\mathbf{z}, \mathcal{H}} := \operatorname{argmin}_{f \in \mathcal{H}} \mathcal{E}_{\mathbf{z}}(f), \quad \text{with} \quad \mathcal{E}_{\mathbf{z}}(f) := \frac{1}{m} \sum_{i=1}^m (y_i - f(x_i))^2. \tag{3}$$

Typically, $\mathcal{H} = \mathcal{H}_m$ depends on a finite number $N = N(m)$ of parameters. In many cases, the number N is chosen using an a priori assumption on f_ρ . In other procedures, the number N is adapted to the

data and thereby avoids any a priori assumptions. We shall be interested in estimators of the latter type.

The usual way of evaluating the performance of the estimator $f_{\mathbf{z}}$ is by studying its convergence either in probability or in expectation, i.e. the rate of decay of the quantities

$$\text{Prob}\{\|f_{\rho} - f_{\mathbf{z}}\| \geq \eta\}, \quad \eta > 0 \quad \text{or} \quad E(\|f_{\rho} - f_{\mathbf{z}}\|^2) \tag{4}$$

as the sample size m increases. Here both the expectation and the probability are taken with respect to the product measure ρ^m defined on Z^m . An estimation of the above probability will automatically give an estimate in expectation by integrating with respect to η . Estimates for the decay of the quantities in (4) are usually obtained under certain assumptions (called *priors*) on f_{ρ} .

It is important to note that the measure ρ_X which appears in the norm (2) is unknown and that we want to avoid any assumption on this measure. This type of regression problem is referred to as *distribution-free*. A recent survey on distribution free regression theory is provided in the book by Györfy et al. (2002), which includes most existing approaches as well as the analysis of their rate of convergence in the expectation sense.

Priors on f_{ρ} are typically expressed by a condition of the type $f_{\rho} \in \Theta$ where Θ is a class of functions that necessarily must be contained in $L_2(X, \rho_X)$. If we wish the error, as measured in (4), to tend to zero as the number m of samples tends to infinity then we necessarily need that Θ is a compact subset of $L_2(X, \rho_X)$. There are three common ways to measure the compactness of a set Θ : (i) minimal coverings, (ii) smoothness conditions on the elements of Θ , (iii) the rate of approximation of the elements of Θ by a specific approximation process. In the learning problem, each of these approaches has to deal with the fact that ρ_X is unknown.

To describe approach (i), for a given Banach space \mathcal{B} which contains Θ , we define the entropy number $\varepsilon_n(\Theta, \mathcal{B})$, $n = 1, 2, \dots$ as the minimal ε such that Θ can be covered by at most 2^n balls of radius ε in \mathcal{B} . The set Θ is compact in $L_2(X, \rho_X)$ if and only if $\varepsilon_n(\Theta, L_2(X, \rho_X))$ tends to zero as $n \rightarrow \infty$. One can therefore quantify the level of compactness of Θ by an assumption on the rate of decay of $\varepsilon_n(\Theta, L_2(X, \rho_X))$. A typical prior condition would be to assume that the entropy numbers satisfy

$$\varepsilon_n(\Theta, \mathcal{B}) \leq Cn^{-r}, \quad n = 1, 2, \dots \tag{5}$$

for some $r > 0$.

Coverings and entropy numbers have a long history in statistics for deriving optimal bounds for the rate of decay in statistical estimation (see e.g. Birgé and Massart, 2001). Several recent works (Cucker and Smale, 2001; DeVore et al., 2004b; Konyagin and Temlyakov, 2004b) have used this technique to bound the error for the regression problem in learning. It has been communicated to us by Lucien Birgé that one can derive from one of his forthcoming papers (Birgé, 2004) that for any class Θ satisfying (5) with $\mathcal{B} = L_2(X, \rho_X)$, there is an estimator $f_{\mathbf{z}}$ satisfying

$$E(\|f_{\rho} - f_{\mathbf{z}}\|^2) \leq Cm^{-\frac{2r}{2r+1}}, \quad m = 1, 2, \dots \tag{6}$$

whenever $f_{\rho} \in \Theta$. Lower bounds which match (6) have been given by DeVore et al. (2004b) using a slightly different type of entropy.

The estimators constructed using this approach are made through ε nets and are more of theoretical interest (in giving the best possible bounds) but are not practical since ρ_X is unknown and therefore these ε nets are also unknown. Another deficiency in this approach is that the estimator typically requires the knowledge of the prior class Θ . One would like to avoid knowledge of Θ in

the construction of an estimator since we do not know f_ρ and hence would generally not have any information about Θ . One can also use ε nets to give bounds for $\text{Prob}(\|f_\rho - f_z\|)$. This is one of the main points in the paper by Cucker and Smale (2001) and is carried further in several other papers (see DeVore et al., 2004b; Konyagin and Temlyakov, 2004a,b).

One way to circumvent the problem of not knowing the marginal ρ_X is to use coverings in the space $C(X)$ of continuous functions equipped with the uniform norm $\|\cdot\|_{L_\infty}$ rather than in $L_2(X, \rho_X)$, since a good covering for Θ in $C(X)$ gives bounds for the covering in $L_2(X, \rho_X)$ independently of ρ_X . In this approach one would assume that Θ satisfies (5) for $\mathcal{B} = C(X)$ and then build estimators which satisfy (6) using ε nets for $C(X)$. Again this does not lead to practical estimators. But the main deficiency of this approach is that the assumption that Θ is a compact subset of $C(X)$ is too severe and does not give a full spectrum of compact subsets of $L_2(X, \rho_X)$.

Concerning (ii), it is well known that when ρ_X is the Lebesgue measure, the unit ball of the Sobolev space $W^r(L_p)$ is a compact set of L_2 under the condition that $\frac{s}{d} > \frac{1}{p} - \frac{1}{2}$. We recall that when r is an integer, $W^r(L_p)$ consists of all L_p functions which distributional derivatives of order $|\alpha| \leq r$ are also in L_p . It is a Banach space when equipped with the norm

$$\|f\|_{W^r(L_p)} := \sup_{|\alpha| \leq r} \|D^\alpha f\|_{L_p}.$$

Similar remarks hold for Sobolev spaces with non-integer r , as well as for the Besov spaces $B_q^r(L_p)$ which offer a more refined description of the notion of r -differentiability in L_p . We refer to DeVore (1998) for the precise definition of such spaces.

However, there is no general approach to defining smoothness spaces with respect to general Borel measures ρ_X which precludes the direct use of classification according to (ii). One way to circumvent this is to define smoothness in $C(X)$, that is systematically use the spaces $W^r(L_\infty)$, but then this suffers from the same deficiency of not giving a full array of compact subsets in $L_2(X, \rho_X)$.

The classification of compactness according to approximation properties (iii) begins with a specific method of approximation and then defines the classes Θ in terms of a rate of approximation by the specified method. The simplest example is to take a sequence (S_n) of linear spaces of dimension n and define Θ as the class of all functions f in $L_2(X, \rho_X)$ which satisfy

$$\inf_{g \in S_n} \|f - g\| \leq C\alpha_n$$

where C is a fixed constant and (α_n) is a sequence of positive real numbers tending to zero. Natural choices for this sequence are $\alpha_n = n^{-r}$, where $r > 0$. Classes defined in such a way will not give a full spectrum of compact subsets in $L_2(X, \rho_X)$. But this deficiency can be removed by using nonlinear spaces Σ_n in place of the linear spaces S_n (see the discussion in DeVore et al., 2004b). An illustrative example is approximation by piecewise polynomials on partitions. If the partitions are set in advance this corresponds to the linear space approximation above. In nonlinear methods the partitions are allowed to vary but their size is specified. We discuss this in more detail later in this paper. An in depth discussion of the approximation theory approach to building estimators for the regression problem in learning is given by DeVore et al. (2004b) and the follow up papers (Konyagin and Temlyakov, 2004a,b).

We should mention that in classical settings, for example when ρ_X is Lebesgue measure then the three approaches to measuring compactness are closely related and in a certain sense equivalent. This is the main chapter of approximation theory.

Concrete algorithms have been constructed for the regression problem in learning by using approximation from specific linear spaces such as piecewise polynomial on uniform partitions, convolution kernels, and spline functions. The rate of convergence of the estimators built from such a linear approximation process is related to the approximation rate of the corresponding process on the class Θ .

A very useful method for bounding the performance of such estimators is provided by the following result (see Györfy et al., 2002, Theorem 11.3): if \mathcal{H} is taken as a linear space of dimension N and if the least-square estimator (3) is post-processed by application of the truncation operator $y \mapsto T_M(y) = \text{sign}(y) \min\{|y|, M\}$, then

$$E(\|f_\rho - f_{\mathbf{z}}\|^2) \leq C \frac{N \log(m)}{m} + \inf_{g \in \mathcal{H}} \|f_\rho - g\|^2.$$

Using this, one can derive specific rates of convergence in expectation by balancing both terms. For example, if Θ is a ball of the Sobolev space $W^r(L_\infty)$ and \mathcal{H} is taken as a space of piecewise polynomial functions of degree no larger than $r - 1$ on uniform partitions of X , one derives

$$E(\|f_\rho - f_{\mathbf{z}}\|^2) \leq C \left(\frac{m}{\log m}\right)^{-\frac{2r}{d+2r}}. \quad (7)$$

This estimate is optimal for this class Θ , up to the logarithmic factor.

The deficiency in this approach is twofold. First, it usually chooses the hypothesis classes in advance and typically assumes knowledge of the prior for this choice. Secondly, it uses linear methods of approximation and therefore misses our goal of giving an estimator which performs optimally for the full range of smoothness spaces in $L_2(X, \rho_X)$.

The first deficiency motivates the notion of *adaptive* or *universal* estimators: the estimation algorithm should be able to exhibit the optimal rate without the knowledge of the exact amount of smoothness r in the regression function f_ρ . A classical way to reach this goal is to perform model selection by adding a complexity regularization term in the empirical risk minimization process (see Barron, 1991; Baraud, 2002; Birgé and Massart, 2001; DeVore et al., 2004b; Györfy et al., 2002, Chapter 12). In particular, one can construct one estimator which simultaneously obtains the optimal rate (7) for all finite balls in each of the class $W^r(L_\infty)$, $0 < r \leq k$ where k is arbitrary but fixed, by the selection of an appropriate uniform partition.

Fixing the second deficiency means that in the case where the marginal ρ_X is Lebesgue measure, the estimator would necessarily have to be optimal for all Sobolev and Besov classes which compactly embed into $L_2(X, \rho_X)$. These spaces correspond to smoothness spaces of order s in L_p whenever $s > \frac{d}{p} - \frac{d}{2}$ (see DeVore, 1998). This can be achieved by introducing spatially adaptive partitions. The selection of an appropriate adaptive partition in the complexity regularization framework can be implemented by the CART algorithm (Breiman et al., 1984), which limits the search within a set of admissible partitions based on a tree structured splitting rule.

A practical limitation of the above described complexity regularization approach is that it is not generally compatible with the practical requirement of *on-line* computations, by which we mean that the estimator for the sample size m can be derived by a simple update of the estimator for the sample size $m - 1$, since the minimization problem needs to be globally re-solved when adding a new sample.

In two slightly different contexts, namely density estimation and denoising on a fixed design, estimation procedures based on *wavelet thresholding* have been proposed as a natural alternative to

model selection by complexity regularization (Donoho and Johnstone, 1998, 1995; Donoho et al., 1996a,b). These procedures are particularly attractive since they combine optimal convergence rates for the largest possible array of unknown priors together with simple and fast algorithms which are on-line implementable. In the learning theory context, the wavelet thresholding has also been used by DeVore et al. (2004a) for estimation of a modification of the regression function f_ρ , namely, for estimating $(d\rho_X/dx)f_\rho$, where ρ_X is assumed to be absolutely continuous with regard to the Lebesgue measure. The main difficulty in generalizing such procedures to the distribution-free regression context is due to the presence of the marginal probability ρ_X in the $L_2(X, \rho_X)$ norm. This typically leads to the need of using wavelet-type bases which are orthogonal (or biorthogonal) with respect to this inner product. Such bases might be not easy to handle numerically and cannot be constructed exactly since ρ_X is unknown.

In this paper, we propose an approach which allows us to circumvent these difficulties, while staying in spirit close to the ideas of wavelet thresholding. In our approach, the hypothesis classes \mathcal{H} are spaces of piecewise constant functions associated to adaptive partitions Λ . Our partitions have the same tree structure as those used in the CART algorithm (Breiman et al., 1984), yet the selection of the appropriate partition is operated quite differently since it is not based on an optimization problem which would have to be re-solved when a new sample is added: instead our algorithm selects the partition through a thresholding procedure applied to empirical quantities computed at each node of the tree which play a role similar to wavelet coefficients. While the connection between CART and thresholding in one or several orthonormal bases is well understood in the fixed design denoising context (Donoho, 1997), this connection is not clear to us in our present context. As it will be demonstrated, our estimation schemes enjoy the following properties:

- (i) They rely on fast algorithms, which may be implemented by simple on-line updates when the sample size m is increased.
- (ii) The error estimates do not require any regularity in $C(X)$ but only in the natural space $L_2(X, \rho_X)$.
- (iii) The proven convergence rates are optimal in probability and expectation (up to logarithmic factors) for the largest possible range of smoothness classes in $L_2(X, \rho_X)$.
- (iv) The scheme is universal in that it does not involve any a-priori knowledge concerning the regularity of f_ρ .

The present choice of piecewise constant functions limits the optimal convergence rate to classes of low or no pointwise regularity. While the practical extension of our method to higher order piecewise polynomial approximations is almost straightforward, its analysis in this more general context becomes significantly more difficult and will be given in a forthcoming paper. This is so far a weakness of our approach from the theoretical perspective, compared to the complexity regularization approach for which optimal convergence results could be obtained in the piecewise polynomial context (using for instance Györfy et al., 2002, Theorem 12.1).

Our paper is organized as follows. The learning algorithm as well as the convergence results are described in Section 2. The next two Sections 3 and 4 are devoted to the proofs of the two main results which deal respectively with the error estimates for non-adaptive and adaptive partitions. Finally, in Section 3 we give results about the consistency of our estimator.

2. The Basic Strategy and the Main Results

In this section we start in §2.1 with some basic facts about adaptive approximation. Then in we continue in §2.2 with some results about least-squares fitting on fixed partition. The universal algorithm is described in §2.3 where the main results of this paper are formulated. In §2.4 we discuss the on-line implementation of our algorithm.

2.1 Partitions and Adaptive Approximation

A typical way of generating partitions Λ of X is through a refinement strategy. We first describe the prototypical example of dyadic partitions. For this, we assume that $X = [0, 1]^d$ and denote by $\mathcal{D}_j = \mathcal{D}_j(X)$ the collection of dyadic subcubes of X of sidelength 2^{-j} and $\mathcal{D} := \cup_{j=0}^{\infty} \mathcal{D}_j$. These cubes are naturally aligned on a tree $\mathcal{T} = \mathcal{T}(\mathcal{D})$. Each node of the tree \mathcal{T} is a cube $I \in \mathcal{D}$. If $I \in \mathcal{D}_j$, then its children are the 2^d dyadic cubes of $J \subset \mathcal{D}_{j+1}$ with $J \subset I$. We denote the set of children of I by $C(I)$. We call I the parent of each such child J and write $I = \mathcal{P}(J)$. The cubes in $\mathcal{D}_j(X)$ form a uniform partition in which every cube has the same measure 2^{-jd} .

More general adaptive partitions are defined as follow. A *proper* subtree $\tilde{\mathcal{T}}$ of \mathcal{T} is a collection of nodes of \mathcal{T} with the properties: (i) the root node $I = X$ is in $\tilde{\mathcal{T}}$, (ii) if $I \neq X$ is in $\tilde{\mathcal{T}}$ then its parent $\mathcal{P}(I)$ is also in $\tilde{\mathcal{T}}$. Any finite proper subtree $\tilde{\mathcal{T}}$ is associated to a unique partition $\Lambda = \Lambda(\tilde{\mathcal{T}})$ which consists of its *outer leaves*, by which we mean those $J \in \mathcal{T}$ such that $J \notin \tilde{\mathcal{T}}$ but $\mathcal{P}(J)$ is in $\tilde{\mathcal{T}}$. One way of generating adaptive partitions is through some refinement strategy. One begins at the root X and decides whether to refine X (i.e. subdivide X) based on some refinement criteria. If X is subdivided then one examines each child and decides whether or not to refine such a child based on the refinement strategy.

The results given in this paper can be described for more general refinement. We shall work in the following setting. We assume that $a \geq 2$ is a fixed integer. We assume that if X is to be refined then its children consist of a subsets of X which are a partition of X . Similarly, for each such child there is a rule which spells out how this child is refined. We assume that the child is also refined into a sets which form a partition of the child. Such a refinement strategy also results in a tree \mathcal{T} (called the *master tree*) and children, parents, proper trees and partitions are defined as above for the special case of dyadic partitions. The refinement level j of a node is the smallest number of refinements (starting at root) to create this node. We denote by \mathcal{T}_j the proper subtree consisting of all nodes with level $< j$ and we denote by Λ_j the partition associated to \mathcal{T}_j , which coincides with $\mathcal{D}_j(X)$ in the above described dyadic partition case. Note that in contrast to this case, the a children may not be similar in which case the partitions Λ_j are not spatially uniform (we could also work with even more generality and allow the number of children to depend on the cell to be refined, while remaining globally bounded by some fixed a). It is important to note that the cardinalities of a proper tree $\tilde{\mathcal{T}}$ and of its associated partition $\Lambda(\tilde{\mathcal{T}})$ are equivalent. In fact one easily checks that

$$\#(\Lambda(\tilde{\mathcal{T}})) = (a - 1)\#(\tilde{\mathcal{T}}) + 1,$$

by remarking that each time a new node gets refined in the process of building an adaptive partition, $\#(\tilde{\mathcal{T}})$ is incremented by 1 and $\#(\Lambda)$ by $a - 1$.

Given a partition Λ , let us denote by \mathcal{S}_Λ the space of piecewise constant functions subordinate to Λ . Each $S \in \mathcal{S}_\Lambda$ can be written

$$S = \sum_{I \in \Lambda} a_I \chi_I,$$

where for $G \subset X$ we denote by χ_G the indicator function, i.e. $\chi_G(x) = 1$ for $x \in G$ and $\chi_G(x) = 0$ for $x \notin G$. We shall consider approximation of a given function $f \in L_2(X, \rho_X)$ by the elements of S_Λ . The best approximation to f in this space is given by

$$P_\Lambda f := \sum_{I \in \Lambda} c_I \chi_I \tag{1}$$

where $c_I = c_I(f)$ is given by

$$c_I := \frac{\alpha_I}{\rho_I}, \text{ with } \alpha_I := \int_I f d\rho_X \text{ and } \rho_I := \rho_X(I). \tag{2}$$

In the case where $\rho_I = 0$, both f_ρ and its projection are undefined on I . For notational reasons, we set in this case $c_I := 0$.

We shall be interested in two types of approximation corresponding to uniform refinement and adaptive refinement. We first discuss uniform refinement. Let

$$E_n(f) := \|f - P_{\Lambda_n} f\|, \quad n = 0, 1, \dots$$

which is the error for uniform refinement. The decay of this error to zero is connected with the smoothness of f as measured in $L_2(X, \rho_X)$. We shall denote by \mathcal{A}^s the approximation class consisting of all functions $f \in L_2(X, \rho_X)$ such that

$$E_n(f) \leq M_0 a^{-ns}, \quad n = 0, 1, \dots \tag{3}$$

Notice that $\#(\Lambda_n) = a^n$ so that the decay in (3) is like N^{-s} with N the number of elements in the partition. The smallest M_0 for which (3) holds serves to define the semi-norm $|f|_{\mathcal{A}^s}$ on \mathcal{A}^s . The space \mathcal{A}^s can be viewed as a smoothness space of order $s > 0$ with smoothness measured with respect to ρ_X .

For example, if ρ_X is the Lebesgue measure and we use dyadic partitioning then $\mathcal{A}^{s/d} = B_\infty^s(L_2)$, $0 < s \leq 1$, with equivalent norms. Here $B_\infty^s(L_2)$ is the Besov space which can be described in terms of differences as

$$\|f(\cdot + h) - f(\cdot)\|_{L_2} \leq M_0 |h|^s, \quad x, h \in X.$$

Instead of working with a-priori fixed partitions there is a second kind of approximation where the partition is generated adaptively and will vary with f . Adaptive partitions are typically generated by using some refinement criterion that determines whether or not to subdivide a given cell. We shall use a refinement criteria that is motivated by adaptive wavelet constructions such as those given by Cohen et al. (2001) for image compression. The criteria we shall use to decide when to refine is analogous to thresholding wavelet coefficients. Indeed, it would be exactly this criteria if we were to construct a wavelet (Haar like) bases for $L_2(X, \rho_X)$.

For each cell I in the master tree \mathcal{T} and any $f \in L_2(X, \rho_X)$ we define

$$\varepsilon_I(f)^2 := \sum_{J \in \mathcal{C}(I)} \frac{\left(\int_J f d\rho_X\right)^2}{\rho_J} - \frac{\left(\int_I f d\rho_X\right)^2}{\rho_I}, \tag{4}$$

which describes the amount of $L_2(X, \rho_X)$ energy which is increased in the projection of f_ρ onto S_Λ when the element I is refined. It also accounts for the decreased projection error when I is refined. In fact, one easily verifies that

$$\varepsilon_I(f)^2 = \|f - c_I\|_{L_2(I, \rho_X)}^2 - \sum_{J \in \mathcal{C}(I)} \|f - c_J\|_{L_2(J, \rho_X)}^2.$$

If we were in a classical situation of Lebesgue measure and dyadic refinement, then $\varepsilon_I(f)^2$ would be exactly the sum of squares of the Haar coefficients of f corresponding to I .

We can use $\varepsilon_I(f)$ to generate an adaptive partition. Given any $\eta > 0$, we let $\mathcal{T}(f, \eta)$ be the smallest proper tree that contains all $I \in \mathcal{T}$ for which $\varepsilon_I(f) \geq \eta$. This tree can also be described as the set of all $J \in \mathcal{T}$ such that there exists $I \subset J$ such that $\varepsilon_I(f) \geq \eta$. Note that since $f \in L^2(X, \rho_X)$ the set of nodes such that $\varepsilon_I(f) \geq \eta$ is always finite and so is $\mathcal{T}(f, \eta)$. Corresponding to this tree we have the partition $\Lambda(f, \eta)$ consisting of the outer leaves of $\mathcal{T}(f, \eta)$. We shall define some new smoothness spaces \mathcal{B}^s which measure the regularity of a given function f by the size of the tree $\mathcal{T}(f, \eta)$.

Given $s > 0$, we let \mathcal{B}^s be the collection of all $f \in L_2(X, \rho_X)$ such that the following is finite

$$|f|_{\mathcal{B}^s}^p := \sup_{\eta > 0} \eta^p \#(\mathcal{T}(f, \eta)), \quad \text{where } p := (s + 1/2)^{-1} \quad (5)$$

We obtain the norm for \mathcal{B}^s by adding $\|f\|$ to $|f|_{\mathcal{B}^s}$. One can show that

$$\|f - P_{\Lambda(f, \eta)}\| \leq C_s |f|_{\mathcal{B}^s}^{\frac{1}{2s+1}} \eta^{\frac{2s}{2s+1}} \leq C_s |f|_{\mathcal{B}^s} N^{-s}, \quad N := \#(\mathcal{T}(f, \eta)), \quad (6)$$

where the constant C_s depends only on s . For the proof of this fact we refer the reader to the paper by Cohen et al. (2001) where a similar result is proven for dyadic partitioning. It follows that every function $f \in \mathcal{B}^s$ can be approximated to order $O(N^{-s})$ by $P_\Lambda f$ for some partition Λ with $\#(\Lambda) = N$. This should be contrasted with \mathcal{A}^s which has the same approximation order for the uniform partition. It is easy to see that \mathcal{B}^s is larger than \mathcal{A}^s . In classical settings, the class \mathcal{B}^s is well understood. For example, in the case of Lebesgue measure and dyadic partitions we know that each Besov space $B_q^s(L_\tau)$ with $\tau > (s/d + 1/2)^{-1}$ and $0 < q \leq \infty$ arbitrary, is contained in $\mathcal{B}^{s/d}$ (see Cohen et al., 2001). This should be compared with the \mathcal{A}^s where we know that $\mathcal{A}^{s/d} = B_\infty^s(L_2)$ as we have noted earlier.

The distinction between these two forms of approximation is that in the first, the partitions are fixed in advance regardless of f but in the second form the partition can adapt to f .

We have chosen here one particular refinement strategy (based on the size of $\varepsilon_I(f)$) in generating our adaptive partitions. According to (6), it provides optimal convergence rates for the class \mathcal{B}^s . There is actually a slightly better strategy described in the paper by Binev and DeVore (2004) which is guaranteed to give near optimal adaptive partitions (independent of the refinement strategy and hence not necessarily of the above form) for each individual f . We have chosen to stick with the present refinement strategy since it extends easily to empirical data (see §2.2) and it is much easier to analyze the convergence properties of this empirical scheme.

2.2 Least-Squares Fitting on Partitions

We now return to the problem of estimating f_ρ from the given data. We shall use the functions in \mathcal{S}_Λ for this purpose. Let us first observe that

$$P_\Lambda f_\rho = \operatorname{argmin}_{f \in \mathcal{S}_\Lambda} \mathcal{E}(f) = \operatorname{argmin}_{f \in \mathcal{S}_\Lambda} \int_{\mathcal{Z}} (y - f(x))^2 d\rho.$$

Indeed, for all $f \in L_2(X, \rho_X)$ we have

$$\mathcal{E}(f) = \mathcal{E}(f_\rho) + \|f - f_\rho\|^2$$

so that minimizing $\mathcal{E}(f)$ over \mathcal{S}_Λ is the same as minimizing $\|f_\rho - f\|$ over $f \in \mathcal{S}_\Lambda$. Note that $P_\Lambda f_\rho$ is obtained by solving N independent problems $\min_{c \in \mathbb{R}} \int_I (f_\rho - c)^2 d\rho_X$ for each element $I \in \Lambda$.

As in (3) we define the estimator $f_{\mathbf{z}, \Lambda}$ of f_ρ on \mathcal{S}_Λ as the empirical counterpart of $P_\Lambda f_\rho$ obtained as the solution of the least-squares problem

$$f_{\mathbf{z}, \Lambda} := \operatorname{argmin}_{f \in \mathcal{S}_\Lambda} \mathcal{E}_{\mathbf{z}}(f) = \operatorname{argmin}_{f \in \mathcal{S}_\Lambda} \frac{1}{m} \sum_{i=1}^m (y_i - f(x_i))^2.$$

We can view our data as a multivalued function y with $y(x_i) = y_i$. Then in analogy to $P_\Lambda f_\rho$, we can view $f_{\mathbf{z}, \Lambda}$ as an orthogonal projection of y onto \mathcal{S}_Λ with respect to the empirical norm

$$\|y\|_{L_2(X, \delta_x)}^2 := \frac{1}{m} \sum_{i=1}^m |y(x_i)|^2,$$

and we can compute it by solving $\#\Lambda$ independent problems

$$\min_{c \in \mathbb{R}} \frac{1}{m} \sum_{i=1}^m (y_i - c)^2 \chi_I(x_i),$$

for each element $I \in \Lambda$. The minimizer $c_I(\mathbf{z})$ is now given by the empirical average

$$c_I(\mathbf{z}) = \frac{\alpha_I(\mathbf{z})}{\rho_I(\mathbf{z})}, \quad \text{where } \alpha_I(\mathbf{z}) := \frac{1}{m} \sum_{i=1}^m y_i \chi_I(x_i), \quad \rho_I(\mathbf{z}) := \frac{1}{m} \sum_{i=1}^m \chi_I(x_i). \quad (7)$$

Thus, we can rewrite the estimator as

$$f_{\mathbf{z}, \Lambda} = \sum_{I \in \Lambda} c_I(\mathbf{z}) \chi_I. \quad (8)$$

In the case where I contains no sample x_i (which may happen even if $\rho_I > 0$), we set $c_I(\mathbf{z}) := 0$.

A natural way of assessing the error $\|f_\rho - f_{\mathbf{z}, \Lambda}\|$ is by splitting it into a bias and stochastic part : since $f_\rho - P_\Lambda f_\rho$ is orthogonal to \mathcal{S}_Λ ,

$$\|f_\rho - f_{\mathbf{z}, \Lambda}\|^2 = \|f_\rho - P_\Lambda f_\rho\|^2 + \|P_\Lambda f_\rho - f_{\mathbf{z}, \Lambda}\|^2 =: e_1 + e_2.$$

Concerning the variance term e_2 , we shall establish the following probability estimate.

Theorem 1 For any partition Λ and any $\eta > 0$,

$$\text{Prob} \left\{ \|P_\Lambda f_\rho - f_{\mathbf{z}, \Lambda}\| > \eta \right\} \leq 4Ne^{-c \frac{m\eta^2}{N}}, \quad (9)$$

where $N := \#(\Lambda)$ and c depends only on M .

As will be explained later in detail, the following estimate of the variance term in expectation is obtained by integration of (9) over $\eta > 0$.

Corollary 1 If Λ is any partition, the mean square error is bounded by

$$E \left(\|P_\Lambda f_\rho - f_{\mathbf{z}, \Lambda}\|^2 \right) \leq C \frac{N \log N}{m}, \quad (10)$$

where $N := \#(\Lambda)$ and the constant C depends only on M .

Let us consider now the case of uniform refinement. We can equilibrate the bias term with the variance term described by Theorem 1 and Corollary 1 and obtain the following result.

Theorem 2 Assume that $f_\rho \in \mathcal{A}^s$ and define the estimator $f_{\mathbf{z}} := f_{\mathbf{z}, \Lambda_j}$ with j chosen as the smallest integer such that $a^{j(1+2s)} \geq \frac{m}{\log m}$. Then, given any $\beta > 0$, there is a constant $\tilde{c} = \tilde{c}(M, \beta, a)$ such that

$$\text{Prob} \left\{ \|f_\rho - f_{\mathbf{z}}\| > (\tilde{c} + |f_\rho|_{\mathcal{A}^s}) \left(\frac{\log m}{m} \right)^{\frac{s}{2s+1}} \right\} \leq Cm^{-\beta}, \quad (11)$$

and

$$E \left(\|f_\rho - f_{\mathbf{z}}\|^2 \right) \leq (C + |f_\rho|_{\mathcal{A}^s}^2) \left(\frac{\log m}{m} \right)^{\frac{2s}{2s+1}}. \quad (12)$$

where C depends only on a and M .

Remark 1 It is also possible to prove Corollary 1 using the result by of Cucker and Smale (2001, Theorem C*). The expectation estimate (12) in Theorem 2 can also be obtained as a consequence of Theorem 11.3 by Györfy et al. (2002) quoted in our introduction. In order to prepare for the subsequent developments direct proofs of these results are given later in §3.

Theorem 2 is satisfactory in the sense that it is obtained under no assumption on the measure ρ_X and the assumption $f_\rho \in \mathcal{A}^s$ is measuring smoothness (and hence compactness) in $L_2(X, \rho_X)$, i.e. the compactness assumption is done in $L_2(\rho_X)$ rather than in L_∞ . Moreover, the rate $\left(\frac{m}{\log m} \right)^{-\frac{s}{2s+1}}$ is known to be optimal (or minimax) over the class \mathcal{A}^s save for the logarithmic factor. However, it is unsatisfactory in the sense that the estimation procedure requires the a-priori knowledge of the smoothness parameter s which appears in the choice of the resolution level j . Moreover, as noted before, the smoothness assumption $f_\rho \in \mathcal{A}^s$ is too severe.

In the context of density estimation or denoising, it is well known that adaptive methods based on wavelet thresholding (Donoho and Johnstone, 1998, 1995; Donoho et al., 1996a,b) allow one to treat both defects. Our next goal is to define similar strategies in our learning context, in which two specific features have to be taken into account : the error is measured in the norm $L_2(X, \rho_X)$ and the marginal probability measure ρ_X is unknown.

2.3 A Universal Algorithm Based on Adaptive Partitions

The main feature of our algorithm is to adaptively choose a partition $\Lambda = \Lambda(\mathbf{z})$ depending on the data \mathbf{z} . It will not require a priori knowledge of the smoothness of f_ρ but rather will learn the smoothness from the data. Thus, it will automatically choose the right size for the partition Λ .

Our starting point is the adaptive procedure introduced in §2.1 applied to the function f_ρ . We use the notation $\varepsilon_I := \varepsilon_I(f_\rho)$ in this case. Then, by (4),

$$\varepsilon_I^2 := \sum_{J \in \mathcal{C}(I)} \frac{\alpha_J^2}{\rho_J} - \frac{\alpha_I^2}{\rho_I}.$$

The selection of the partition Λ in our learning scheme will be based on the empirical coefficients

$$\varepsilon_I^2(\mathbf{z}) := \sum_{J \in \mathcal{C}(I)} \frac{\alpha_J^2(\mathbf{z})}{\rho_J(\mathbf{z})} - \frac{\alpha_I^2(\mathbf{z})}{\rho_I(\mathbf{z})}.$$

We define the threshold

$$\tau_m := \kappa \sqrt{\frac{\log m}{m}}, \tag{13}$$

where the constant κ is absolute and will be fixed later in the proof of Theorem 3 stated below. Let $\gamma > 0$ be an arbitrary but fixed constant. We define $j_0 = j_0(m, \gamma)$ as the largest integer j such that $a^j \leq \tau_m^{-1/\gamma}$. We next consider the smallest proper tree $\mathcal{T}(\mathbf{z}, m)$ which contains the set

$$\Sigma(\mathbf{z}, m) := \{I \in \mathcal{T}_{j_0} ; \varepsilon_I(\mathbf{z}) \geq \tau_m\}.$$

This tree can also be described as the set of all $J \in \mathcal{T}_{j_0}$ such that there exists $I \subset J$ such that $I \in \Sigma(\mathbf{z}, m)$. We then define the partition $\Lambda = \Lambda(\mathbf{z}, m)$ associated to this tree and the corresponding estimator $f_{\mathbf{z}} := f_{\mathbf{z}, \Lambda}$. In summary, our algorithm consists in the following steps:

- (i) Compute the $\varepsilon_I(\mathbf{z})$ for the nodes I of generation $j < j_0$.
- (ii) Threshold these quantities at level τ_m to obtain the set $\Sigma(\mathbf{z}, m)$.
- (iii) Complete $\Sigma(\mathbf{z}, m)$ to $\mathcal{T}(\mathbf{z}, m)$ by adding the nodes J which contain an $I \in \Sigma(\mathbf{z}, m)$.
- (iv) Compute the estimator $f_{\mathbf{z}}$ by empirical risk minimization on the partition $\Lambda(\mathbf{z}, m)$.

Further comments on the implementation will be given in the next section. The main result of this paper is the following theorem.

Theorem 3 *Let $\beta, \gamma > 0$ be arbitrary. Then, there exists $\kappa_0 = \kappa_0(\beta, \gamma, M)$ such that if $\kappa \geq \kappa_0$, then whenever $f_\rho \in \mathcal{A}^\gamma \cap \mathcal{B}^s$ for some $s > 0$, the following concentration estimate holds*

$$\text{Prob} \left\{ \|f_\rho - f_{\mathbf{z}}\| \geq \tilde{c} \left(\frac{\log m}{m} \right)^{\frac{s}{2s+1}} \right\} \leq Cm^{-\beta}, \tag{14}$$

as well as the following expectation bound

$$E(\|f_\rho - f_{\mathbf{z}}\|^2) \leq C \left(\frac{\log m}{m} \right)^{\frac{2s}{2s+1}}, \tag{15}$$

where the constants \tilde{c} and C are independent of m .

Theorem 3 is more satisfactory than Theorem 2 in two respects: (i) the optimal rate $(\frac{\log m}{m})^{\frac{s}{2s+1}}$ is now obtained under weaker smoothness assumptions on the regression function, namely, $f_\rho \in \mathcal{B}^s$ in place of $f_\rho \in \mathcal{A}^s$, with the extra assumption of $f_\rho \in \mathcal{A}^\gamma$ smoothness with $\gamma > 0$ arbitrarily small, (ii) the algorithm is universal. Namely, the value of s does not enter the definition of the algorithm. Indeed, the algorithm automatically exploits this unknown smoothness through the samples \mathbf{z} . We note however that the algorithm does require the knowledge of the parameter γ which can be arbitrarily small. It is actually possible to build an algorithm without assuming knowledge of a $\gamma > 0$ by using the adaptive tree algorithm by Binev and DeVore (2004). However, the implementation of such an algorithm would involve complications we wish to avoid in this presentation.

2.4 Remarks on Algorithmic Aspects and On-Line Implementation

Our first remarks concern the construction of the adaptive partition $\Lambda(\mathbf{z}, m)$ for a fixed m which requires the computation of the numbers $\varepsilon_I(\mathbf{z})$ for $I \in \Lambda_j$ when j satisfies $a^j \leq \tau_m^{-1/\gamma}$. This would require the computation of $O(m \ln m)$ coefficients. One can actually save a substantial amount of computation by remarking that by definition we always have

$$\varepsilon_I(\mathbf{z})^2 \leq \mathcal{E}_I(\mathbf{z})$$

with $\mathcal{E}_I(\mathbf{z}) := \|y - c_I(\mathbf{z})\|_{L_2(I, \delta_x)}^2$ the least-square error on I . In contrast to $\varepsilon_I(\mathbf{z})$, the quantity $\mathcal{E}_I(\mathbf{z})$ is monotone with respect to inclusion:

$$J \subset I \Rightarrow \mathcal{E}_J(\mathbf{z}) \leq \mathcal{E}_I(\mathbf{z}).$$

This allows one to organize the search for those I satisfying $\varepsilon_I(\mathbf{z}) \geq \tau_m$ from coarse to fine elements. In particular, one no longer has to check those descendants of an element I for which $\mathcal{E}_I(\mathbf{z})$ is less than τ_m .

Our next remarks concern the on-line implementation of the algorithm. Suppose that we have computed $\rho_I(\mathbf{z})$, $\alpha_I(\mathbf{z})$ and the $\varepsilon_I(\mathbf{z})$ where \mathbf{z} contains m samples. If we now add a new sample (x_{m+1}, y_{m+1}) to \mathbf{z} to obtain \mathbf{z}^+ , the new ρ_I and α_I are

$$\rho_I(\mathbf{z}^+) = \frac{m}{m+1}(\rho_I(\mathbf{z}) + \chi_I(x_{m+1}))$$

and

$$\alpha_I(\mathbf{z}^+) = \frac{m}{m+1}(\alpha_I(\mathbf{z}) + y_{m+1}\chi_I(x_{m+1})).$$

In particular, we see that at each level j , only one I is affected by the new sample. Therefore, if we store the quantities $\rho_I(\mathbf{z})$ and $\alpha_I(\mathbf{z})$ in the current partition, then this new step requires at most j_0 additional computations in the case where j_0 is not increased. In the case where j_0 is increased to $j_0 + 1$ (this may happen because τ_m is decreased), the computations of the quantities $\rho_I(\mathbf{z})$ and $\alpha_I(\mathbf{z})$ need to be performed, of course, for all the elements in the newly added level.

3. Proof of the Results on Non-Adaptive Partitions

We first give the proof of Theorem 1. Let Λ be any partition. By (1) and (8), we can write

$$\|P_\Lambda f_\rho - f_{\Lambda, \mathbf{z}}\|^2 = \sum_{I \in \Lambda} |c_I - c_I(\mathbf{z})|^2 \rho_I.$$

According to their definitions (2), (7), both c_I and $c_I(\mathbf{z})$ are bounded in modulus by M . Therefore, given $\eta > 0$, if we define

$$\Lambda^- := \{I \in \Lambda : \rho_I \leq \frac{\eta^2}{8NM^2}\},$$

we clearly have

$$\sum_{I \in \Lambda^-} |c_I - c_I(\mathbf{z})|^2 \rho_I \leq \frac{\eta^2}{2}.$$

We next consider the complement set $\Lambda^+ = \Lambda \setminus \Lambda^-$. In order to prove (9), it now suffices to establish that for all $I \in \Lambda^+$

$$\text{Prob} \left\{ |c_I(\mathbf{z}) - c_I|^2 \geq \frac{\eta^2}{2N\rho_I} \right\} \leq 4e^{-c\frac{m\eta^2}{N}}. \quad (1)$$

To see this, we write $\rho_I(\mathbf{z}) = (1 + \mu_I)\rho_I$ and remark that if $|\mu_I| \leq 1/2$ we have

$$\begin{aligned} |c_I(\mathbf{z}) - c_I| &= \left| \frac{\alpha_I(\mathbf{z})}{\rho_I(\mathbf{z})} - \frac{\alpha_I}{\rho_I} \right| = \frac{1}{\rho_I(1 + \mu_I)} |\alpha_I(\mathbf{z}) - \alpha_I - \mu_I \alpha_I| \\ &\leq 2\rho_I^{-1} (|\alpha_I(\mathbf{z}) - \alpha_I| + |\alpha_I \mu_I|). \end{aligned}$$

It follows that $|c_I(\mathbf{z}) - c_I| \leq \frac{\eta}{\sqrt{2N\rho_I}}$ provided that we have jointly

$$|\alpha_I(\mathbf{z}) - \alpha_I| \leq \frac{\eta\sqrt{\rho_I}}{4\sqrt{2N}},$$

and (since $\alpha_I \mu_I = \alpha_I(\rho_I(\mathbf{z}) - \rho_I)/\rho_I$)

$$|\rho_I(\mathbf{z}) - \rho_I| \leq \min \left\{ \frac{1}{2}\rho_I, \frac{\eta\rho_I^{3/2}}{4\sqrt{2N}|\alpha_I|} \right\}$$

and therefore

$$\begin{aligned} \text{Prob} \left\{ |c_I(\mathbf{z}) - c_I|^2 \geq \frac{\eta^2}{2N\rho_I} \right\} &\leq \text{Prob} \left\{ |\alpha_I(\mathbf{z}) - \alpha_I| \geq \frac{\eta\sqrt{\rho_I}}{4\sqrt{2N}} \right\} \\ &+ \text{Prob} \left\{ |\rho_I(\mathbf{z}) - \rho_I| \geq \min \left\{ \frac{1}{2}\rho_I, \frac{\eta\rho_I^{3/2}}{4\sqrt{2N}|\alpha_I|} \right\} \right\}. \end{aligned}$$

In order to estimate these probabilities, we shall use Bernstein's inequality which says that for m independent realizations ζ_i of a random variable ζ such that $|\zeta(z) - E(\zeta)| \leq M_0$ and $\text{Var}(\zeta) = \sigma^2$, one has for any $\varepsilon > 0$

$$\text{Prob} \left\{ \left| \frac{1}{m} \sum_{i=1}^m \zeta_i - E(\zeta) \right| \geq \varepsilon \right\} \leq 2e^{-\frac{m\varepsilon^2}{2(\sigma^2 + M_0\varepsilon/3)}}.$$

In our context, we apply this inequality to $\zeta = y\chi_I(x)$ for which $E(\zeta) = \alpha_I$, $M_0 \leq 2M$ and $\sigma^2 \leq M^2\rho_I$, and to $\zeta = \chi_I(x)$ for which $E(\zeta) = \rho_I$, $M_0 \leq 1$, and $\sigma^2 \leq \rho_I$.

We first obtain that

$$\begin{aligned} \text{Prob} \left\{ |\alpha_I(\mathbf{z}) - \alpha_I| \geq \frac{\eta\sqrt{\rho_I}}{4\sqrt{2N}} \right\} &\leq 2e^{-\frac{m\eta^2\rho_I}{64N(M^2\rho_I+2M\eta\sqrt{\rho_I/2N/12})}} \\ &\leq 2e^{-\frac{m\eta^2\rho_I}{64N(M^2\rho_I+4M^2\rho_I/12)}} \\ &\leq 2e^{-c\frac{m\eta^2}{N}}, \end{aligned}$$

with $c = [\frac{256}{3}M^2]^{-1}$, where we have used in the second inequality that $I \in \Lambda^+$ to bound the second term in the denominator of the exponential by the first term in the denominator. We next obtain in the case where $\frac{1}{2}\rho_I \leq \frac{\eta\rho_I^{3/2}}{4\sqrt{2N}|\alpha_I|}$

$$\text{Prob} \left\{ |\rho_I(\mathbf{z}) - \rho_I| \geq \frac{1}{2}\rho_I \right\} \leq 2e^{-\frac{m\rho_I^2}{8(\rho_I+\rho_I/6)}} = 2e^{-\frac{3}{28}m\rho_I} \leq 2e^{-c\frac{m\eta^2}{N}}$$

with $c = [\frac{224}{3}M^2]^{-1}$ where we have used in the last line that $I \in \Lambda^+$. Finally, in the case where $\frac{1}{2}\rho_I \geq \frac{\eta\rho_I^{3/2}}{4\sqrt{2N}|\alpha_I|}$, we obtain

$$\text{Prob} \left\{ |\rho_I(\mathbf{z}) - \rho_I| \geq \frac{\eta\rho_I^{3/2}}{4\sqrt{2N}|\alpha_I|} \right\} \leq 2e^{-\frac{m\eta^2\rho_I^3}{64N\rho_I|\alpha_I|^2(7\rho_I/6)}} \leq 2e^{-c\frac{m\eta^2}{N}}$$

with $c = [\frac{448}{6}M^2]^{-1}$ since $|\alpha_I| \leq M\rho_I$. Therefore, we obtain (1) with the smallest of the three values of c , namely $c = [\frac{256}{3}M^2]^{-1}$, which concludes the proof of Theorem 1.

Remark 2 *The constant c in the estimate behaves like $1/M^2$ and therefore degenerates to 0 as $M \rightarrow +\infty$. This is due to the fact that we are using Bernstein's estimate as a concentration inequality since we are lacking any other information on the conditional law $\rho(y|x)$. For more specific models where we have more information on the conditional law $\rho(y|x)$, one can avoid the limitation $|y| \leq M$. For instance, in the Gaussian regression problem $y_i = f_\rho(x_i) + g_i$ where g_i are i.i.d. Gaussian (and therefore unbounded) variables $\mathcal{N}(0, \sigma^2)$, the probabilistic estimate (9) can be obtained by a direct use of the concentration property of the Gaussian.*

The proof of Corollary 1 follows by integration of (9) over η :

$$\begin{aligned} E \left(\|P_\Lambda f_\rho - f_{\Lambda, \mathbf{z}}\|_{L_2(X, \rho_X)}^2 \right) &= \int_0^{+\infty} \eta \text{Prob} \left\{ \|P_\Lambda f_\rho - f_{\Lambda, \mathbf{z}}\|_{L_2(\rho_X)} > \eta \right\} d\eta \\ &\leq \int_0^{+\infty} \eta \min \left\{ 1, 4Ne^{-c\frac{m\eta^2}{N}} \right\} d\eta \\ &= \int_0^{\eta_0} \eta d\eta + \int_{\eta_0}^{+\infty} 4N\eta e^{-c\frac{m\eta^2}{N}} d\eta \\ &= \frac{\eta_0^2}{2} + \frac{2N^2}{cm} e^{-c\frac{m\eta_0^2}{N}}, \end{aligned}$$

where η_0 is such that $4Ne^{-c\frac{m\eta_0^2}{N}} = 1$, or equivalently $\eta_0^2 = \frac{N \log(4N)}{cm}$. This proves the estimate (10).

Finally, to prove the estimates in Theorem 2, we first note that, by assumption, $N = \#(\Lambda_j) \leq a^{j+1} \leq a^2 \left(\frac{m}{\log m}\right)^{\frac{1}{2s+1}}$. Further, from the definition of \mathcal{A}^s , we have

$$\|f_\rho - P_{\Lambda_j f_\rho}\| \leq |f_\rho|_{\mathcal{A}^s} a^{-js} \leq |f_\rho|_{\mathcal{A}^s} \left(\frac{\log m}{m}\right)^{\frac{s}{2s+1}}.$$

Hence, using Theorem 1, we see that the probability on the left of (11) is bounded from above by

$$\text{Prob} \left\{ \|P_{\Lambda} f_\rho - f_{\Lambda, \mathbf{z}}\| > \tilde{c} \left(\frac{\log m}{m}\right)^{\frac{s}{2s+1}} \right\} \leq 4a^2 m e^{-\frac{c\tilde{c}^2 \log m}{a^2}}$$

which does not exceed $Cm^{-\beta}$ provided $\tilde{c}^2 c > a^2(1 + \beta)$. The proof of (12) follows in a similar way from Corollary 1.

4. Proof of Theorem 3

This section is devoted to a proof of Theorem 3. We begin with our notation. Recall that the tree $\mathcal{T}(f_\rho, \eta)$ is the smallest tree which contains all I for which $\varepsilon_I = \varepsilon_I(f_\rho)$ is larger than η . $\Lambda(f_\rho, \eta)$ is the partition induced by the outer leaves of $\mathcal{T}(f_\rho, \eta)$. We use τ_m as defined in (13) and $j_0 = j_0(m)$ is the largest integer such that $a^{j_0} \leq \tau_m^{-1/\gamma}$. For any partition Λ we write $f_{\mathbf{z}, \Lambda} = \sum_{I \in \Lambda} c_I(\mathbf{z}) \chi_I$.

If Λ_0 and Λ_1 are two adaptive partitions respectively associated to trees \mathcal{T}_0 and \mathcal{T}_1 we denote by $\Lambda_0 \vee \Lambda_1$ and $\Lambda_0 \wedge \Lambda_1$ the partitions associated to the trees $\mathcal{T}_0 \cup \mathcal{T}_1$ and $\mathcal{T}_0 \cap \mathcal{T}_1$, respectively. Given any $\eta > 0$, we define the partitions $\Lambda(\eta) := \Lambda(f_\rho, \eta) \wedge \Lambda_{j_0}$ and $\Lambda(\eta, \mathbf{z})$ associated with the smallest trees containing those I such that $\varepsilon_I \geq \eta$ and $\varepsilon_I(\mathbf{z}) \geq \eta$, respectively, and such that the refinement level j of any I in either one of these two partitions satisfies $j \leq j_0$. In these terms our estimator $f_{\mathbf{z}}$ is given by

$$f_{\mathbf{z}} = f_{\mathbf{z}, m} = f_{\mathbf{z}, \Lambda(\tau_m, \mathbf{z})}.$$

With this notation in hand, we begin now with the proof of the Theorem. Using the triangle inequality, we have

$$\|f_\rho - f_{\mathbf{z}, m}\| \leq e_1 + e_2 + e_3 + e_4$$

with each term defined by

$$\begin{aligned} e_1 &:= \|f_\rho - P_{\Lambda(\tau_m, \mathbf{z}) \vee \Lambda(b\tau_m)} f_\rho\|, \\ e_2 &:= \|P_{\Lambda(\tau_m, \mathbf{z}) \vee \Lambda(b\tau_m)} f_\rho - P_{\Lambda(\tau_m, \mathbf{z}) \wedge \Lambda(\tau_m/b)} f_\rho\|, \\ e_3 &:= \|P_{\Lambda(\tau_m, \mathbf{z}) \wedge \Lambda(\tau_m/b)} f_\rho - f_{\mathbf{z}, \Lambda(\tau_m, \mathbf{z}) \wedge \Lambda(\tau_m/b)}\|, \\ e_4 &:= \|f_{\mathbf{z}, \Lambda(\tau_m, \mathbf{z}) \wedge \Lambda(\tau_m/b)} - f_{\mathbf{z}, \Lambda(\tau_m, \mathbf{z})}\|, \end{aligned}$$

with $b := 2\sqrt{a-1} > 1$. This type of splitting is classically used in the analysis of wavelet thresholding procedures, in order to deal with the fact that the partition built from those I such that $\varepsilon_I(\mathbf{z}) \geq \tau_m$ does not exactly coincides with the partition which would be chosen by an oracle based on those I such that $\varepsilon_I \geq \tau_m$. This is accounted by the terms e_2 and e_4 which correspond to those I such that $\varepsilon_I(\mathbf{z})$ is significantly larger or smaller than ε_I respectively, and which will be proved to be small in probability. The remaining terms e_1 and e_3 respectively correspond to the bias and variance of oracle estimators based on partitions obtained by thresholding the unknown coefficients ε_I .

The first term e_1 is therefore treated by a deterministic estimate. Namely, since $\Lambda(\tau_m, \mathbf{z}) \vee \Lambda(b\tau_m)$ is a finer partition than $\Lambda(b\tau_m)$, we have with probability one

$$\begin{aligned} e_1 &\leq \|f_\rho - P_{\Lambda(b\tau_m)}f_\rho\| \leq \|f_\rho - P_{\Lambda(f_\rho, b\tau_m)}f_\rho\| + \|P_{\Lambda(f_\rho, b\tau_m)}f_\rho - P_{\Lambda(b\tau_m)}f_\rho\| \\ &\leq \|f_\rho - P_{\Lambda(f_\rho, b\tau_m)}f_\rho\| + \|f_\rho - P_{\Lambda_{j_0}}f_\rho\| \\ &\leq C_s(b\tau_m)^{\frac{2s}{2s+1}}|f_\rho|_{\mathcal{B}^s} + a^{-\gamma j_0}|f_\rho|_{\mathcal{A}^\gamma} \\ &\leq C_s(b\tau_m)^{\frac{2s}{2s+1}}|f_\rho|_{\mathcal{B}^s} + a^\gamma \tau_m |f_\rho|_{\mathcal{A}^\gamma}. \end{aligned}$$

Therefore we conclude that

$$e_1 \leq C_s((b\kappa)^{\frac{2s}{2s+1}} + a^\gamma \kappa) \max\{|f_\rho|_{\mathcal{A}^\gamma}, |f_\rho|_{\mathcal{B}^s}\} \left(\frac{\log m}{m}\right)^{\frac{s}{2s+1}}, \quad (1)$$

whenever $f \in \mathcal{B}^s \cap \mathcal{A}^\gamma$.

The third term e_3 is treated by the estimate (9) of Theorem 1:

$$\text{Prob}\{e_3 > \eta\} \leq 4Ne^{-c\frac{m\eta^2}{N}}, \quad (2)$$

with

$$N = \#\{\Lambda(\tau_m, \mathbf{z}) \wedge \Lambda(\tau_m/b)\} \leq \#\{\Lambda(\tau_m/b)\} \leq \#\{\Lambda(f_\rho, \tau_m/b)\}.$$

Hence we infer from (5) that

$$N \leq b^p \tau_m^{-p} |f_\rho|_{\mathcal{B}^s}^p = b^p \tau_m^{-\frac{2}{2s+1}} |f_\rho|_{\mathcal{B}^s}^p = b^p \kappa^{-\frac{2}{2s+1}} |f_\rho|_{\mathcal{B}^s}^p \left(\frac{m}{\log m}\right)^{\frac{1}{2s+1}}, \quad (3)$$

where we have used that $1/p = 1/2 + s$.

Concerning the two remaining terms e_2 and e_4 , we shall prove that for a fixed but arbitrary $\beta > 0$, we have

$$\text{Prob}\{e_2 > 0\} + \text{Prob}\{e_4 > 0\} \leq Cm^{-\beta}, \quad (4)$$

whenever $\kappa \geq \kappa_0$ with κ_0 depending on β , γ , and M and with C depending only on a .

Before proving this result, let us show that the combination (1), (2), (3) and (4) imply the validity of the estimates (14) and (15) in Theorem 3. We fix the value of β and we fix any constant κ for which (4) holds. Let $\eta_1 := \tilde{c}(\frac{\log m}{m})^{\frac{s}{2s+1}}$ with \tilde{c} from (14) and $\eta_2 := c_0(\frac{\log m}{m})^{\frac{s}{2s+1}}$ with $c_0 := C_s(\kappa^{\frac{2s}{2s+1}} + a^\gamma \kappa) \max\{|f_\rho|_{\mathcal{A}^\gamma}, |f_\rho|_{\mathcal{B}^s}\}$. From (1) it follows that for $\tilde{c} > c_0$ we have $\text{Prob}\{\|f_\rho - f_{\mathbf{z}, m}\| > \eta_1\} \leq \text{Prob}\{e_2 + e_3 + e_4 > \eta_1 - \eta_2\}$. Hence, defining $\eta = (\tilde{c} - c_0)(\frac{\log m}{m})^{\frac{s}{2s+1}}$, the probability on the left side of (14) does not exceed

$$\text{Prob}\{e_2 > 0\} + \text{Prob}\{e_3 > \eta\} + \text{Prob}\{e_4 > 0\} \leq \text{Prob}\{e_3 > \eta\} + Cm^{-\beta},$$

Moreover, on account of (2) and (3), we can estimate $\text{Prob}\{e_3 > \eta\}$ by

$$\begin{aligned} \text{Prob}\{e_3 > \eta\} &\leq C \left(\frac{m}{\log m}\right)^{\frac{1}{2s+1}} e^{-cm\eta^2 b^{-p} \kappa^{-\frac{2}{2s+1}} |f_\rho|_{\mathcal{B}^s}^{-p}} \left(\frac{\log m}{m}\right)^{\frac{1}{2s+1}} \\ &= C \left(\frac{m}{\log m}\right)^{\frac{1}{2s+1}} e^{-cD^2 m \left(\frac{\log m}{m}\right)} \\ &= C \left(\frac{m}{\log m}\right)^{\frac{1}{2s+1}} m^{-cD^2} \\ &\leq Cm^{1-cD^2} \end{aligned}$$

where $D^2 := \frac{(\tilde{c}-c_0)^2}{\kappa^{2s+1} b^p |f|_{\mathcal{B}^s}^p}$. The concentration estimate (14) follows now by taking \tilde{c} large enough so that $1 - cD^2 + \beta \leq 0$.

For the expectation estimate (15), we recall that according to Corollary 1, we have

$$E(e_3^2) \leq C \frac{N \log N}{m} \leq C \frac{\left(\frac{m}{\log m}\right)^{\frac{1}{2s+1}} \log m}{m} = C \left(\frac{\log m}{m}\right)^{\frac{2s}{1+2s}}.$$

We then remark that we always have $e_2^2 \leq 4M^2$, and therefore

$$E(e_2^2) \leq 4M^2 \text{Prob}\{e_2 > 0\} \leq C m^{-\beta} \leq C \left(\frac{m}{\log m}\right)^{-\frac{2s}{1+2s}},$$

by choosing β larger than $2s/(2s+1)$, for example $\beta = 1$. The same holds for e_4 and therefore we obtain (15).

It remains to prove (4). The main tool here is a probabilistic estimate of how the empirical coefficient $\varepsilon_I(\mathbf{z})$ may differ from ε_I with respect to the threshold. This is expressed by the following lemma.

Lemma 4 *For any $\eta > 0$ and any element $I \in \mathcal{T}$, one has*

$$\text{Prob}\{\varepsilon_I(\mathbf{z}) \leq \eta \text{ and } \varepsilon_I \geq b\eta\} \leq C e^{-c m \eta^2} \quad (5)$$

and

$$\text{Prob}\{\varepsilon_I \leq \eta \text{ and } \varepsilon_I(\mathbf{z}) \geq b\eta\} \leq C e^{-c m \eta^2} \quad (6)$$

where the constant c depends only on M and the constant C depends only on a .

Before proving Lemma 4, let us show how this result implies (4). We first consider the second term e_2 . Clearly $e_2 = 0$ if $\Lambda(\tau_m, \mathbf{z}) \vee \Lambda(b\tau_m) = \Lambda(\tau_m, \mathbf{z}) \wedge \Lambda(\tau_m/b)$ or equivalently $\mathcal{T}(\tau_m, \mathbf{z}) \cup \mathcal{T}(b\tau_m) = \mathcal{T}(\tau_m, \mathbf{z}) \cap \mathcal{T}(\tau_m/b)$. Now if the inclusion $\mathcal{T}(\tau_m, \mathbf{z}) \cap \mathcal{T}(\tau_m/b) \subset \mathcal{T}(\tau_m, \mathbf{z}) \cup \mathcal{T}(b\tau_m)$ is strict, then one either has $\mathcal{T}(\tau_m, \mathbf{z}) \not\subset \mathcal{T}(\tau_m/b)$ or $\mathcal{T}(b\tau_m) \not\subset \mathcal{T}(\tau_m, \mathbf{z})$. Thus, there either exists an I such that both $\varepsilon_I(\mathbf{z}) < \tau_m$ and $\varepsilon_I \geq b\tau_m$ or there exists an I such that both $\varepsilon_I(\mathbf{z}) \geq \tau_m$ and $\varepsilon_I < \tau_m/b$. It follows that

$$\begin{aligned} \text{Prob}\{e_2 > 0\} &\leq \sum_{I \in \mathcal{T}_{j_0}} \text{Prob}\{\varepsilon_I(\mathbf{z}) \leq \tau_m \text{ and } \varepsilon_I \geq b\tau_m\} \\ &\quad + \sum_{I \in \mathcal{T}_{j_0}} \text{Prob}\{\varepsilon_I(\mathbf{z}) \geq \tau_m \text{ and } \varepsilon_I \leq \tau_m/b\}. \end{aligned} \quad (7)$$

Using (5) with $\eta = \tau_m$ yields

$$\begin{aligned} \sum_{I \in \mathcal{T}_{j_0}} \text{Prob}\{\varepsilon_I(\mathbf{z}) \leq \tau_m \text{ and } \varepsilon_I \geq b\tau_m\} &\leq \#(\mathcal{T}_{j_0}) e^{-c m \tau_m^2} \\ &\leq \#(\Lambda_{j_0}) e^{-c m \tau_m^2} \\ &\leq a^{j_0} e^{-c \kappa^2 \log m} \\ &\leq \tau_m^{-1/\gamma} m^{-c \kappa^2} \\ &\leq C m^{1/\gamma - c \kappa^2}. \end{aligned}$$

We can treat the second sum in (7) the same way and obtain the same bound as the one for e_4 below. By similar considerations, we obtain

$$\text{Prob}\{e_4 > 0\} \leq \sum_{I \in \mathcal{T}_{j_0}} \text{Prob}\{\varepsilon_I(\mathbf{z}) \geq \tau_m \text{ and } \varepsilon_I \leq \tau_m/b\},$$

and we use (6) with $\eta = \tau_m/b$ which yields $\text{Prob}\{e_4 > 0\} \leq Cm^{1/\gamma - c\kappa^2/b^2}$. We therefore obtain (4) by choosing $\kappa \geq \kappa_0$ with $c\kappa_0^2 = b^2(\beta + 1/\gamma)$.

We are left with the proof of Lemma 4. As a first step, we show that the proof can be reduced to the particular case $a = 2$. To this end, we remark that the splitting of I into its a children $\{J_1, \dots, J_a\}$ can be decomposed into $a - 1$ steps consisting of splitting an element into a pair of elements: defining $I_n := I \setminus (J_1 \cup \dots \cup J_n)$ we start from $I = I_0$ and refine iteratively I_{n-1} into the two elements I_n and J_n , for $n = 1, \dots, a - 1$. By orthogonality, we can write

$$\varepsilon_I^2 := \sum_{n=0}^{a-2} (\varepsilon_{I_n})^2,$$

where $\varepsilon_{I_n}^2$ is the amount of $L_2(X, \rho_X)$ energy which is increased in the projection of f_ρ when I_{n+1} is refined into I_n and J_n . In a similar way, we can write for the observed quantities

$$\varepsilon_I^2(\mathbf{z}) := \sum_{n=0}^{a-2} \varepsilon_{I_n}^2(\mathbf{z}),$$

Now if $\varepsilon_I^2 \leq \eta^2$ and $\varepsilon_I(\mathbf{z})^2 \geq b^2\eta^2 = 4(a-1)\eta^2$, it follows that there exist $n \in \{0, \dots, a-2\}$ such that $(\varepsilon_{I_n})^2 \leq \eta^2$ and $\varepsilon_{I_n}(\mathbf{z})^2 \geq 4\eta^2$. Therefore,

$$\text{Prob}\{\varepsilon_I \leq \eta \text{ and } \varepsilon_I(\mathbf{z}) \geq b\eta\} \leq \sum_{n=0}^{a-2} \text{Prob}\{\varepsilon_{I_n} \leq \eta \text{ and } \varepsilon_{I_n}(\mathbf{z}) \geq 2\eta\},$$

and similarly

$$\text{Prob}\{\varepsilon_I(\mathbf{z}) \leq \eta \text{ and } \varepsilon_I \geq b\eta\} \leq \sum_{n=0}^{a-2} \text{Prob}\{\varepsilon_{I_n}(\mathbf{z}) \leq \eta \text{ and } \varepsilon_{I_n} \geq 2\eta\},$$

so that the estimates (5) and (6) for $a > 2$ follow from the same estimates established for $a = 2$ in which case $b = 2$.

In the case $a = 2$, we denote by I^+ and I^- the two children of I . Note that if $\rho_J = 0$ for $J = I^+$ or for $J = I^-$, there is nothing to prove, since in this case we find that $\varepsilon_I = \varepsilon_I(\mathbf{z}) = 0$ with probability one. We therefore assume that $\rho_J > 0$ for $J = I^+$ and I^- . We first rewrite ε_I as follows

$$\begin{aligned} \varepsilon_I^2 &= \frac{\alpha_{I^+}^2}{\rho_{I^+}} + \frac{\alpha_{I^-}^2}{\rho_{I^-}} - \frac{\alpha_I^2}{\rho_I} = \rho_{I^+}c_{I^+}^2 + \rho_{I^-}c_{I^-}^2 - \rho_Ic_I^2 \\ &= \rho_{I^+}c_{I^+}^2 + \rho_{I^-}c_{I^-}^2 - \rho_I((\rho_{I^+}c_{I^+} + \rho_{I^-}c_{I^-})/\rho_I)^2 \\ &= \frac{\rho_{I^+}\rho_{I^-}}{\rho_I}(c_{I^+} - c_{I^-})^2, \end{aligned}$$

and therefore $\varepsilon_I = |\beta_I|$ with

$$\beta_I := \sqrt{\frac{\rho_{I^+}\rho_{I^-}}{\rho_I}}(c_{I^+} - c_{I^-}).$$

In a similar way we obtain $\varepsilon_I(\mathbf{z}) = |\beta_I(\mathbf{z})|$ with

$$\beta_I(\mathbf{z}) := \sqrt{\frac{\rho_{I^+}(\mathbf{z})\rho_{I^-}(\mathbf{z})}{\rho_I(\mathbf{z})}}(c_{I^+}(\mathbf{z}) - c_{I^-}(\mathbf{z})).$$

Introducing the quantities $a_{I^+} = \sqrt{\frac{\rho_{I^-}}{\rho_I\rho_{I^+}}}$ and $a_{I^-} = \sqrt{\frac{\rho_{I^+}}{\rho_I\rho_{I^-}}}$ and their empirical counterpart $a_{I^+}(\mathbf{z})$ and $a_{I^-}(\mathbf{z})$ we can rewrite β_I and $\beta_I(\mathbf{z})$ as

$$\beta_I = a_{I^+}\alpha_{I^+} - a_{I^-}\alpha_{I^-}$$

and

$$\beta_I(\mathbf{z}) = a_{I^+}(\mathbf{z})\alpha_{I^+}(\mathbf{z}) - a_{I^-}(\mathbf{z})\alpha_{I^-}(\mathbf{z}).$$

It follows that

$$|\varepsilon_I - \varepsilon_I(\mathbf{z})| \leq |a_{I^+}\alpha_{I^+} - a_{I^+}(\mathbf{z})\alpha_{I^+}(\mathbf{z})| + |a_{I^-}\alpha_{I^-} - a_{I^-}(\mathbf{z})\alpha_{I^-}(\mathbf{z})|.$$

We next introduce the numbers δ_J defined by the relation $\rho_J(\mathbf{z}) = (1 + \delta_J)\rho_J$, for $J = I^+, I^-$ or I . It is easily seen that if $|\delta_J| \leq \delta \leq 1/4$ for $J = I^+, I^-$ and I , one has

$$a_{I^+}(\mathbf{z}) = (1 + \mu_I^+)a_{I^+}$$

with $|\mu_I^+| \leq 3\delta$. This follows indeed from the basic inequalities

$$1 - 3\delta \leq \sqrt{\frac{(1-\delta)}{(1+\delta)^2}} \leq \sqrt{\frac{(1+\delta)}{(1-\delta)^2}} \leq 1 + 3\delta$$

which hold for $0 \leq \delta \leq 1/4$. Therefore if $|\delta_J| \leq \delta \leq 1/4$ for $J = I^+, I^-$ and I , we have

$$\begin{aligned} |a_{I^+}\alpha_{I^+} - a_{I^+}(\mathbf{z})\alpha_{I^+}(\mathbf{z})| &\leq a_{I^+}(\mathbf{z})|\alpha_{I^+} - \alpha_{I^+}(\mathbf{z})| + |\alpha_{I^+}(a_{I^+} - a_{I^+}(\mathbf{z}))| \\ &\leq 2a_{I^+}|\alpha_{I^+} - \alpha_{I^+}(\mathbf{z})| + 3\delta a_{I^+}|\alpha_{I^+}|. \end{aligned}$$

By similar considerations, we obtain the estimate

$$|a_{I^-}\alpha_{I^-} - a_{I^-}(\mathbf{z})\alpha_{I^-}(\mathbf{z})| \leq 2a_{I^-}|\alpha_{I^-} - \alpha_{I^-}(\mathbf{z})| + 3\delta a_{I^-}|\alpha_{I^-}|,$$

and therefore

$$|\varepsilon_I - \varepsilon_I(\mathbf{z})| \leq \sum_{K=I^+, I^-} 2a_K|\alpha_K - \alpha_K(\mathbf{z})| + 3\delta a_K|\alpha_K|. \quad (8)$$

We first turn to (5), which corresponds to the case where $\varepsilon_I \geq 2\eta$ and $\varepsilon_I(\mathbf{z}) \leq \eta$. In this case, we remark that we have

$$\eta^2 \leq \frac{\varepsilon_I^2}{4} = \frac{\rho_{I^+}\rho_{I^-}}{\rho_I} \frac{(c_{I^+} - c_{I^-})^2}{4} \leq M^2\rho_L, \quad (9)$$

for $L = I^+, I^-$ and I . Combining (8) and (9), we estimate the probability by

$$\text{Prob}\{\varepsilon_I(\mathbf{z}) \leq \eta \text{ and } \varepsilon_I \geq 2\eta\} \leq \sum_{K=I^+, I^-} (p_K + \sum_{J=I^-, I^+, I} q_{K,J}), \quad (10)$$

with

$$p_K := \text{Prob}\{|\alpha_K - \alpha_K(\mathbf{z})| \geq [8a_K]^{-1}\eta \text{ given } \rho_K \geq \frac{\eta^2}{M^2}\},$$

and

$$q_{K,J} := \text{Prob}\{|\rho_J - \rho_J(\mathbf{z})| \geq \rho_J \min\{\frac{1}{4}, \eta[12a_K|\alpha_K|]^{-1}\} \text{ given } \rho_J \geq \frac{\eta^2}{M^2}\}.$$

Using Bernstein's inequality, we can estimate p_K as follows

$$p_K \leq 2e^{-\frac{m\eta^2}{2(64a_K^2M^2\rho_K+8a_K\eta M/3)}} \leq 2e^{-\frac{m\eta^2}{2(64a_K^2M^2\rho_K+8a_K\sqrt{\rho_K}M^2/3)}} \leq 2e^{-cm\eta^2},$$

with $c = [(128 + 16/3)M^2]^{-1}$, where we have used $\eta^2 \leq \rho_K M^2$ in the second inequality and the fact that $a_K^2 \rho_K \leq 1$ in the third inequality.

In the case where $12a_K|\alpha_K| \leq 4\eta$, we estimate $q_{K,J}$ by

$$q_{K,J} \leq 2e^{-\frac{m\rho_J}{2(16+4/3)}} \leq 2e^{-cm\eta^2},$$

with $c = [(32 + 8/3)M^2]^{-1}$, where we have used $\rho_J \geq \eta^2/M^2$.

In the opposite case $12a_K|\alpha_K| \geq 4\eta$, we estimate $q_{K,J}$ by

$$q_{K,J} \leq 2e^{-m \frac{\left(\frac{\rho_J \eta}{12a_K|\alpha_K|}\right)^2}{2\left(\rho_J + \frac{\rho_J \eta}{36a_K|\alpha_K|}\right)}} \leq 2e^{-\frac{m\rho_J \eta^2}{312a_K^2|\alpha_K|^2}}$$

where in the last inequality we used $3a_K|\alpha_K| \geq \eta$ to bound the second term in the denominator. Since $|\alpha_K| \leq M\rho_K$, we have $a_K^2 \alpha_K^2 \leq M^2(\rho_{I^-} \rho_{I^+} / \rho_I) \leq M^2 \min\{\rho_{I^-}, \rho_{I^+}\}$ so that $\rho_J \geq a_K^2 \alpha_K^2 / M^2$. Therefore, we obtain

$$q_{K,J} \leq e^{-cm\eta^2}$$

with $c = [312M^2]^{-1}$.

Using these estimates for p_K and $q_{K,J}$ back in (10), we obtain (5).

We next turn to (6), which corresponds to the opposite case where $\varepsilon_I \leq \eta$ and $\varepsilon_I(\mathbf{z}) \geq 2\eta$. In this case, we remark that we have

$$\eta^2 \leq \frac{\varepsilon_I^2(\mathbf{z})}{4} = \frac{\rho_{I^+}(\mathbf{z})\rho_{I^-}(\mathbf{z})}{\rho_I(\mathbf{z})} \frac{(c_{I^+}(\mathbf{z}) - c_{I^-}(\mathbf{z}))^2}{4} \leq M^2 \rho_L(\mathbf{z}),$$

for $L = I^+, I^-$ and I . In this case, we do not have $\eta^2 \leq M^2 \rho_L$, but we shall use the fact that $\eta^2 \leq 2M^2 \rho_L$ with high probability, by writing

$$\text{Prob}\{\varepsilon_I \leq \eta \text{ and } \varepsilon_I(\mathbf{z}) \geq 2\eta\} \leq \sum_{K=I^+, I^-, I} (p_K + \tilde{p}_K + \sum_{J=I^-, I^+, I} (q_{K,J} + \tilde{p}_J)), \quad (11)$$

where now

$$p_K := \text{Prob}\{|\alpha_K - \alpha_K(\mathbf{z})| \geq [8a_K]^{-1}\eta; \text{ given } \rho_K \geq \frac{\eta^2}{2M^2}\},$$

and

$$q_{K,J} := \text{Prob}\{|\rho_J - \rho_J(\mathbf{z})| \geq \rho_J \min\{\frac{1}{4}, \eta[12a_K|\alpha_K|]^{-1}\} \text{ given } \rho_J \geq \frac{\eta^2}{2M^2}\}$$

and the additional probability is given by

$$\tilde{p}_J := \text{Prob}\{\eta^2 \leq M^2\rho_J(\mathbf{z}) \text{ given } \eta^2 \geq 2M^2\rho_J\}.$$

Clearly, p_K and $q_{K,J}$ are estimated as in the proof of (5). The additional probability is estimated by

$$\begin{aligned} \tilde{p}_J &\leq \text{Prob}\{\eta^2 \geq M^2\rho_J \text{ and } |\rho_J - \rho_J(\mathbf{z})| \geq (\eta/M)^2\} \\ &\leq 2e^{-\frac{m\eta^4}{2(\rho_J M^4 + M^2\eta/3)}} \\ &\leq 2e^{-\frac{m\eta^4}{2(\eta^2 M^2 + M^2\eta^2/3)}} \\ &\leq 2e^{-cm\eta^2}, \end{aligned}$$

with $c = (8M^2/3)^{-1}$. Using these estimates in (11), we obtain (6), which concludes the proof of the lemma. \square

5. Universal Consistency of the Estimator

In this last section, we discuss the consistency of our estimator when no smoothness assumption is made on the regression function $f_\rho \in L^2(X, \rho_X)$. Of course it is still assumed that $|y| \leq M$ almost surely, so that we also have $|f_\rho| \leq M$. For an arbitrary such f_ρ , we are interested in proving the convergence property

$$\lim_{m \rightarrow +\infty} E(\|f_\rho - f_{\mathbf{z},m}\|^2) = 0,$$

which in turn implies the convergence in probability: for all $\varepsilon > 0$,

$$\lim_{m \rightarrow +\infty} \text{Prob}\{\|f_\rho - f_{\mathbf{z},m}\| > \varepsilon\} = 0.$$

For this purpose, we use the same estimation of the error by $e_1 + e_2 + e_3 + e_4$ as in the proof of Theorem 3.

We first remark that the proof of the estimate

$$E(e_2^2) + E(e_4^2) \leq Cm^{-\beta},$$

remains unchanged under no smoothness assumption made on f_ρ .

Concerning the approximation term e_1 , we have seen that

$$e_1 \leq \|f_\rho - P_{\Lambda(f_\rho, b\tau_m)} f_\rho\| + \|f_\rho - P_{\Lambda_{j_0}} f_\rho\|.$$

Under no smoothness assumptions, the convergence to 0 of these two terms still occurs when $j_0 \rightarrow +\infty$ and $\tau_m \rightarrow 0$, and therefore as $m \rightarrow +\infty$. This requires however that the union of the spaces $(S_{\Lambda_j})_{j \geq 0}$ is dense in $L^2(X, \rho_X)$. This is ensured by imposing natural restrictions on the splitting procedure generating the partitions which should be such that

$$\lim_{j \rightarrow +\infty} \sup_{I \in \Lambda_j} |I| = 0,$$

where $|I|$ is the Lebesgue measure of I . This is obviously true for dyadic partitions, and more generally when the splitting rule is such that

$$\sum_{J \in \mathcal{C}(I)} |J| \leq \nu |I|,$$

with $\nu < 1$ independent of $I \in \mathcal{T}$. Under this restriction, classical results of measure theory state that $P_{\Lambda_j} f$ converges to f in $L^2(X, \rho_X)$ as $j \rightarrow +\infty$ for all $f \in L^2(\rho_X)$.

We are therefore ensured that $\|f_\rho - P_{\Lambda_{j_0}} f_\rho\|$ tends to 0 as $m \rightarrow +\infty$. For the first term $\|f_\rho - P_{\Lambda(f_\rho, b\tau_m)} f_\rho\|$, we remark that the convergence of $P_{\Lambda_j} f$ to f also implies that f can be written as the sum of an $L^2(X, \rho_X)$ -orthogonal series

$$f = c_X \chi_X + \sum_{I \in \mathcal{T}} \psi_I, \quad \text{with } \psi_I := \sum_{J \in \mathcal{C}(I)} c_J \chi_J - c_I \chi_I,$$

We remark that $\|\psi_I\| = \varepsilon_I(f)$. It follows that for $\eta > 0$

$$\|f - P_{\Lambda(f, \eta)} f\|^2 = \sum_{I \notin \mathcal{T}(f, \eta)} \varepsilon_I(f)^2 \leq \sum_{\varepsilon_I(f) \leq \eta} \varepsilon_I(f)^2.$$

Since by Parseval inequality,

$$\sum_{I \in \mathcal{T}} \varepsilon_I(f)^2 = \|f\|^2 - \|c_X \chi_X\|^2 < +\infty, \quad (1)$$

it follows that $\|f - P_{\Lambda(f, \eta)} f_\rho\|$ tends to 0 as $\eta \rightarrow 0$. Therefore $\|f_\rho - P_{\Lambda(f_\rho, b\tau_m)} f_\rho\|$ tends to 0 as $m \rightarrow +\infty$.

It remains to study the variance term e_3 for which we have established

$$E(e_3^2) \leq C \frac{N \log N}{m},$$

with

$$N = \#(\Lambda(\tau_m, \mathbf{z}) \wedge \Lambda(\tau_m/b)) \leq \#(\Lambda(\tau_m/b)).$$

Note that since $(\varepsilon_I)_{I \in \mathcal{T}}$ is a square summable sequence according to (1), we have

$$\#\{I \in \mathcal{T} ; \varepsilon_I > \eta\} \leq C \eta^{-2} \varphi(\eta),$$

where $\varphi(\eta) \rightarrow 0$ as $\eta \rightarrow 0$. Therefore if $\#(\Lambda(\tau_m/b))$ was simply controlled by $\#\{I \in \mathcal{T} ; \varepsilon_I > \tau_m/b\}$, we would derive that $E(e_3^2)$ would tend to 0 according to

$$E(e_3^2) \leq C \frac{\tau_m^{-2} \varphi(\tau_m) \log(\tau_m^{-2} \varphi(\tau_m))}{m} \leq \tilde{C} \frac{\tau_m^{-2} \varphi(\tau_m) \log m}{m} = \tilde{C} \varphi(\tau_m).$$

However, $\#(\Lambda(\tau_m/b))$ can be significantly larger due to the process of completing the set of thresholded coefficients into a proper tree. Since this process adds at most $j_0 - 1$ nodes J for each I such that $\varepsilon_I > \tau_m/b$, we have the estimate

$$\#(\Lambda(\tau_m/b)) \leq j_0 \#\{I \in \mathcal{T} ; \varepsilon_I > \tau_m/b\} \leq C \tau_m^{-2} \varphi(\tau_m) \log m,$$

where C depends on a and γ . It follows that if the threshold τ_m is modified into

$$\tau_m := \frac{\log m}{\sqrt{m}},$$

we find that $E(e_3^2)$ goes to 0 according to

$$E(e_3^2) \leq C \frac{\tau_m^{-2} \varphi(\tau_m) \log m \log(\tau_m^{-2} \varphi(\tau_m) \log m)}{m} \leq \tilde{C} \frac{\tau_m^{-2} \varphi(\tau_m) \log m}{m} = \tilde{C} \varphi(\tau_m).$$

It is easily checked that this modification does not affect the other estimates for e_1 , e_2 and e_4 . However it induces an additional $\sqrt{\log m}$ factor in the rate of convergence which was obtained in Theorem 3.

An alternate way of ensuring the convergence to zero of $E(e_3^2)$ is by imposing that $\gamma > 1/2$, since we obviously have

$$\#(\Lambda(\tau_m/b)) \leq \#(\Lambda_{j_0}) = a^{j_0} \leq C \tau_m^{-1/\gamma},$$

so that $N \log N/m$ tends to 0 if $1/\gamma > 2$. However this is a stronger restriction since the optimal convergence rate of the algorithm is maintained only for regression functions which are at least in the uniform approximation space $\mathcal{A}^{1/2}$.

Acknowledgments

This research was supported in part by the Office of Naval Research Contracts ONR-N00014-03-1-0051, ONR-N00014-03-1-0675 and ONR-N00014-00-1-0470; the Army Research Office Contract DAAD 19-02-1-0028; the AFOSR Contract UF/USAF F49620-03-1-0381; the NSF contracts DMS-0221642 and DMS-0200187; and EEC Human Potential Programme under contract HPRN-CT-2002-00286, “Breaking Complexity”.

References

- Y. Baraud. Model selection for regression on a random design. *ESAIM Prob. et Stats.*, 6:127—146, 2002.
- A. R. Barron. Complexity regularization with application to artificial neural network. In *Nonparametric functional estimation and related topics*, G. Roussas (ed.), pages 561—576, Kluwer Academic Publishers, 1991.
- P. Binev and R. DeVore. Fast computation in adaptive tree approximation. *Numerische Math.*, 97:193—217, 2004.
- L. Birgé. Model selection via testing : an alternative to (penalized) maximum likelihood estimators. Preprint, to appear in *Ann. IHP*, 2004.
- L. Birgé and P. Massart. Gaussian model selection. *J. Eur. Math. Soc.*, 3:203—268, 2001.
- L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*, Wadsworth international, Belmont, CA, 1984.

- A. Cohen, W. Dahmen, I. Daubechies, and R. DeVore. Tree-structured approximation and optimal encoding. *App. Comp. Harm. Anal.*, 11:192—226, 2001.
- S. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of AMS*, 39:1—49, 2001.
- I. Daubechies. *Ten Lectures on Wavelets*, SIAM, Philadelphia, 1992.
- R. A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51—150, 1998.
- R. DeVore, G. Kerkyacharian, D. Picard and V. Temlyakov. On mathematical methods of learning. *IMI Preprint*, 2004:10, University of South Carolina, 2004a.
- R. DeVore, G. Kerkyacharian, D. Picard and V. Temlyakov. Lower bounds in learning theory. *IMI Preprint*, 2004:22, University of South Carolina, 2004b.
- D. L. Donoho. CART and best-ortho-basis : a connection. *Annals of Statistics*, 25:1870—1911, 1997.
- D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via Wavelet shrinkage. *J. Amer. Statist. Assoc.*, 90(no. 432):1200—1224, 1995.
- D. L. Donoho and I. M. Johnstone. Minimax estimation via wavelet shrinkage. *Annals of Statistics*, 26(no. 3):879—921, 1998.
- D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *Journal of the Royal Statistical Society*, 57:301—369, 1996a.
- D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Density estimation by wavelet thresholding. *Annals of Statistics*, 24:508—539, 1996b.
- L. Györfy, M. Kohler, A. Krzyzak, and H. Walk. *A distribution-free theory of nonparametric regression*, Springer, Berlin, 2002.
- S. V. Konyagin and V. N. Temlyakov. Some error estimates in learning theory. *IMI Preprints*, 2004:05, University of South Carolina, 2004a.
- S. V. Konyagin and V. N. Temlyakov. The entropy in learning theory: Error estimates. *IMI Preprints*, 2004:09, University of South Carolina, 2004b.

Efficient Computation of Gapped Substring Kernels on Large Alphabets

Juho Rousu

*Department of Computer Science
Royal Holloway University of London
Egham, TW20 0EX, United Kingdom*

JUHO@CS.RHUL.AC.UK

John Shawe-Taylor

*School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, United Kingdom*

JST@ECS.SOTON.AC.UK

Editor: Tommi Jaakkola

Abstract

We present a sparse dynamic programming algorithm that, given two strings s and t , a gap penalty λ , and an integer p , computes the value of the gap-weighted length- p subsequences kernel. The algorithm works in time $O(p|M|\log|t|)$, where $M = \{(i, j) | s_i = t_j\}$ is the set of matches of characters in the two sequences. The algorithm is easily adapted to handle bounded length subsequences and different gap-penalty schemes, including penalizing by the total length of gaps and the number of gaps as well as incorporating character-specific match/gap penalties.

The new algorithm is empirically evaluated against a full dynamic programming approach and a trie-based algorithm both on synthetic and newswire article data. Based on the experiments, the full dynamic programming approach is the fastest on short strings, and on long strings if the alphabet is small. On large alphabets, the new sparse dynamic programming algorithm is the most efficient. On medium-sized alphabets the trie-based approach is best if the maximum number of allowed gaps is strongly restricted.

Keywords: kernel methods, string kernels, text categorization, sparse dynamic programming

1. Introduction

Machine learning algorithms working on sequence data are needed both in bioinformatics and text categorization and mining. In contrast, standard machine learning algorithms work on feature vector representation, thus requiring a feature extraction phase to map sequence data into feature vectors.

Representing these feature vectors explicitly is often problematic due to the potentially high dimensionality. Kernel methods (Vapnik, 1995; Cristianini and Shawe-Taylor, 2000) provide an efficient way of tackling the problem of dimensionality via the use of a kernel function, corresponding to the inner product of two feature vectors. With these precomputed inner products, it is possible to efficiently accomplish a variety of machine learning and data analysis tasks, e.g. classification, regression and clustering.

The family of kernel functions defined on feature vectors computed from strings, are called *string kernels* (Watkins, 2000; Haussler, 1999). These kernels are based on features corresponding to occurrences of certain kinds of subsequences in the string. There is a wide variety of string kernels depending on how the subsequences are defined: they can be contiguous or non-contiguous, they

can have bounded or unbounded length, and the mismatches or gaps can be penalized in different ways.

There are three main approaches in computing string kernels efficiently. Dynamic programming approaches (Lodhi et al., 2000; Cancedda et al., 2003) are based on composing the solution from simpler subproblems, in this case, from kernel values of shorter subsequences and prefixes of the two strings. These approaches usually have time complexity of order $\Omega(p|s||t|)$ since one typically needs to compute intermediate results for each character pair s_i, t_j for each subsequence length $1 \leq l \leq p$. However, there is no extra computational cost associated when using gap penalties or mismatch costs between the characters. In trie-based approaches (Leslie et al., 2003; Leslie and Kuang, 2003) one makes a depth-first traversal to an implicit trie data structure. The search continues along each trie path while in both of the strings there exist an occurrence of the p -gram corresponding to the trie node. This termination condition prunes the search space very efficiently if the number of gaps is restricted enough. The third approach is to build a suffix tree of one of the strings and then compute matching statistics of the other string by traversing the suffix tree to compute matching statistics (Vishwanathan and Smola, 2003). The computation of the kernel value takes a linear time. However, the approach does not deal with gapped strings.

In this paper, we concentrate on improving the time-efficiency of the dynamic programming approach to gapped string kernel computation. In Section 2 we review types of kernels that are used in text categorization and sequence analysis tasks. As a full review of kernel based machine learning is not possible in the context of this article, a reader not familiar with kernel methods might want to refer to the introductory text book of Cristianini and Shawe-Taylor (2000) or, for a more broad treatment, the books by Schölkopf and Smola (2001) and Shawe-Taylor and Cristianini (2004). In Section 3, we review trie-based and a dynamic programming approaches for gap-weighted string kernel computation before presenting the main contribution of this article, a sparse dynamic programming algorithm for efficiently computing the kernel on large alphabets. We also discuss variants and implementation of the algorithm. In Section 4 the new algorithm is experimentally compared against the full dynamic programming approach and a trie-based algorithm. Results and open problems are discussed in Section 5 followed by conclusions in Section 6.

2. Kernels for Sequence Data

Kernel methods encompass a family of pattern analysis methods that share a common aspect: mapping the inputs $x \in X$ to some potentially high-dimensional feature space \mathcal{F} by defining a feature map $\phi : X \mapsto \mathcal{F}$, and then solving the pattern analysis task by linear methods, such as finding a separating hyperplane for instances of different classes (support vector machines, SVM), or finding principal components of the feature vectors $\phi(x)$ (kernel PCA), or finding correlations between two views $\phi_1(x), \phi_2(x)$ of the same data (kernel canonical correlation analysis, KCCA). Working in these high-dimensional spaces is made possible by the use of the so called 'kernel trick': one does not need to handle the feature vectors explicitly, as long as the inner product, the *kernel*, $K(x, z) = \phi(x)^T \phi(z)$ has been computed.

For example, support vector machines (Vapnik, 1995) find for the training data $\{(x_i, y_i)\}_{i=1}^{\ell}$ the maximum margin separating hyperplane in the feature space. Both learning the hyperplane and classifying points can be done without explicitly using the feature vectors: learning requires solving

a quadratic programme

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - 1/2 \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \text{ s.t. } \sum_i \alpha_i y_i = 0,$$

and the SVM prediction can be expressed as $f(x) = \text{sign}(\sum_i \alpha_i y_i K(x_i, x) + b)$. Thus, the learning and prediction can be performed in space that has dimension in the order of the number of the training points.

When handling input data that already comes in vector form, there is no obligation to introduce a special kernel function. The inner product of the inputs $K(x, z) = x^T z$, also called the 'linear kernel', can be used. However, when using structured data such as sequences, trees or graphs, one needs to convert the structured representation to a vector form.

For sequences the most common feature representation is to count or check the existence of sub-sequence occurrences, when the subsequences are taken from a fixed index set U . Different choices for the index set and accounting for occurrences give rise to a family of feature representations and kernels. Below we review the main forms of representation for sequences and the computation kernels for such representations.

Word spectrum (Bag-of-words) kernels. In the most widely used feature representation for strings in a natural language, informally called *bag-of-words* (BoW), the index set is taken as the set of words in the language, possibly excluding some frequently occurring stop words (Salton et al., 1975). The representation was brought to SVM learning by Joachims (1998).

In the case of a string s containing English text, for each English word u , we define the feature value

$$\phi_u(s) = |\{j | s_j \dots s_{j+|u|-1} = u\}|, \tag{1}$$

as the number of times u occurs in some position j of s . For the example text $s = \text{'The cat was chased by the fat dog'}$ the BoW will contain the following non-zero entries: $\phi_{\text{the}}(s) = 2$, $\phi_{\text{dog}}(s) = 1$, $\phi_{\text{was}}(s) = 1$, $\phi_{\text{chased}}(s) = 1$, $\phi_{\text{by}}(s) = 1$, $\phi_{\text{fat}}(s) = 1$, $\phi_{\text{cat}} = 1$. These occurrence counts can also be weighted, for example by scaling by the inverse document frequency (TFIDF, Salton & Young, 1973):

$$\phi_u(s) = |\{j | s_j \dots s_{j+|u|-1} = u\}| \times \log_2 N/N_u,$$

where N_u is the number of documents where u occurs and N is the total number of documents in the collection.

Although the dimension of the feature space may be very high, computation of the BoW kernel can be efficiently implemented by scanning the two strings, constructing lists $L(s)$ and $L(t)$ of pairs (u, c_u) of word u and occurrence count c_u ordered in the lexicographical order of the substrings u , and finally traversing the two lists to compute the dot product.

Substring spectrum kernels. For strings that do not encompass a crisply defined word-structure, for example, biological sequences, a different approach is more suitable. Given an alphabet Σ , a simple choice is to take $U = \Sigma^p$, the set of strings of length p . The features $\phi_u(s), u \in \Sigma^p$ are then defined as in (1). For example, if we choose $p = 4$ resulting feature values for our example text include $\phi_{\text{the}} = 2$, $\phi_{\text{the}} = 1$, $\phi_{\text{cat}} = 1$, $\phi_{\text{dog}} = 1$, along with close to thirty additional 4-grams.

There is a two-fold difficulty in focusing in fixed length subsequences: Firstly, one may not know how to best choose the length p . Secondly, there maybe important subsequences of different lengths in the sequences. This problem can be circumvented by allowing the lengths of the

subsequences vary within a range:

$$U = \Sigma^q \cup \Sigma^{q+1} \dots \cup \Sigma^p \text{ for some } 1 \leq q \leq p. \quad (2)$$

We call the resulting kernel the *bounded-length substring* kernel. In our example text, we could set $q = 3$ and $p = 5$ to include features such as ϕ_{dog} , ϕ_{chase} and ϕ_{fat} , for instance. In the extreme case, we can take $p = 1$ and $q = \infty$, thus including in the index set all non-empty sequences of alphabet Σ . It should be noted that the choice of parameters q and p has several effects: First, as will be discussed in the next section, the time to compute the kernels will increase by increasing p . Secondly, if all important subsequences have length at least some q_0 , setting $q < q_0$ will probably make the spectrum more 'noisy'. Similarly, setting $q_0 < q$ will probably lose some of the 'signal'. An interesting direction, that is out of scope of this paper, would be to learn the parameters p and q from the data.

The most efficient approaches, working in $O(p(|s| + |t|))$ time, to compute substring spectrum kernels are based on suffix trees (Leslie et al., 2002; Vishwanathan and Smola, 2003), although dynamic programming and trie-based approaches also can be used.

Gapped substring spectrum kernels. Another way to add flexibility to our feature representation is to allow gaps in the subsequence occurrences. In that case, the index set of (2) can still be used but the definition of the features changes. For convenience of notation, in the following we will use boldface letters to indicate ordered collections of indices: $\mathbf{j} = (j_1, j_2, \dots, j_q), j_1 < j_2 < \dots < j_q$ and denote $s(\mathbf{j}) = s_{j_1} \dots s_{j_q}$. We define the features $\phi_u(s)$ to count the number of such unique sequences of indices \mathbf{j} that the corresponding subsequence $s_{j_1} \dots s_{j_q}$ equals to u , in other words $\phi_u(s) = |\{\mathbf{j} | s(\mathbf{j}) = u\}|$. Our example string 'The cat was chased by the fat dog' can be seen to contain, among others, the following gapped substrings of length 3: $\phi_{tea} = 7, \phi_{ted} = 5, \phi_{dog} = 2$.

This definition does not make a difference between occurrences that contain few gaps and those that contain several gaps, or the lengths thereof; all contribute to the feature value equally. For example, the substring 'tea' will have a high weight as it occurs many times in the text, although it never occurs with fewer than two gaps and the total length of gaps is at least three. At the same time, 'dog' will have much smaller weight although it occurs in the text without any gaps.

A solution for this problem is to downweight occurrences that contain many or long gaps. Such feature representation is the basis of *gap-weighted string kernels*. In string matching, there are many approaches for weighting gaps (see e.g. Eppstein, 1989; Gordon et al., 2003). We consider two gap-weighting schemes, both of which downweight occurrences exponentially in increasing gap number or length.

When downweighting by the total length of gaps the weight of an occurrence $\mathbf{i} = (i_1, \dots, i_q)$ with span $span(\mathbf{i}) = i_q - i_1 + 1$ is defined as $\lambda^{span(\mathbf{i})}$, where $0 < \lambda \leq 1$ is a fixed penalty constant. The feature values are then defined as a normalized sum of occurrence weights

$$\phi_u(s) = 1/\lambda^q \sum_{\mathbf{i}: u=s(\mathbf{i})} \lambda^{span(\mathbf{i})}.$$

The normalization $1/\lambda^q$ ensures that only gaps—not matches—are penalized. This normalization is important when using substrings of different lengths as the index set \mathcal{U} , otherwise short substrings easily get too much weight.

In some applications the actual length of the gaps may not be important but the number of contiguous substrings that compose the gapped subsequence may be more relevant. The features to

be computed will then be

$$\phi_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{\sum_j \mathbb{1}[i_{j+1}-i_j>1]}, u \in \Sigma^p,$$

where the expression inside the indicator function $\mathbb{1}[\cdot]$ counts pairs of adjacent indices (i_j, i_{j+1}) within \mathbf{i} where there's a gap in between the two indices.

In literature, two approaches for computing gapped substring kernels appear, a dynamic programming approach (Lodhi et al., 2002) and a trie-based approach (Leslie et al., 2002), both of which can deal with gap-weighting as well. We review the dynamic programming approach and a variant of the trie-based computation in Section 3, followed by a presentation of the new sparse dynamic programming approach.

Generalized alphabets. We conclude this section by noting that treating text as strings of characters is not the only and not necessarily the best approach. Depending on the application, considering larger units such as syllables (c.f. Saunders et al., 2002) or words (c.f. Cancedda et al., 2003) may be beneficial. Using the text 'The cat was chased by the fat dog' as the example, if words are used as the alphabet:

- Substrings are word sequences, or phrases: 'cat was chased'.
- Gapped substrings will be phrases with some words skipped: 'cat ? chased ? ? ? dog'.
- Penalizing gaps will decrease the weight of phrases that span too long a text segment. For example, the weight of 'cat was chased' would be higher than that of 'cat ? chased ? ? ? dog' as the former phrase exists in the text as such whereas the latter contains two gaps of total length four.

Using phrases as features has a potential advantage over the bag-of-words representation, as the ordering and the proximity of the words is taken into account. Thus such a representation should be able to more closely capture syntactically and, ideally, semantically similar text segments.

There is, however, one drawback in using words or phrases as the features, namely the slight variations in the word occurrences that still correspond to the same meaning. Such variations include alternative spellings, prefixes/suffixes attached to words or word stems. These problems can of course be tackled by preprocessing the documents. An alternative approach is the use of syllable alphabet: the text is treated as a sequence of syllables: 'The cat was cha sed by the fat dog'. The benefit is that small spelling variations or inflection of the word (e.g. 'cha se' vs. 'cha sed') are likely to retain some of the original syllables.

Compared to the character alphabet, word and syllable alphabets share two benefits. Firstly, as argued above, using phrases of words or syllables are more likely to capture meaning in the text than arbitrary substring of characters. Secondly, as the document size drastically goes down when moving from character to syllable and word alphabets, computational requirements decrease as well.

3. Computing Gap-Weighted String Kernels

Let us now concentrate to the question how to efficiently compute the gap-weighted string kernel:

$$K(s, t) = \sum_{u \in U} \phi_u(s) \phi_u(t), \tag{3}$$

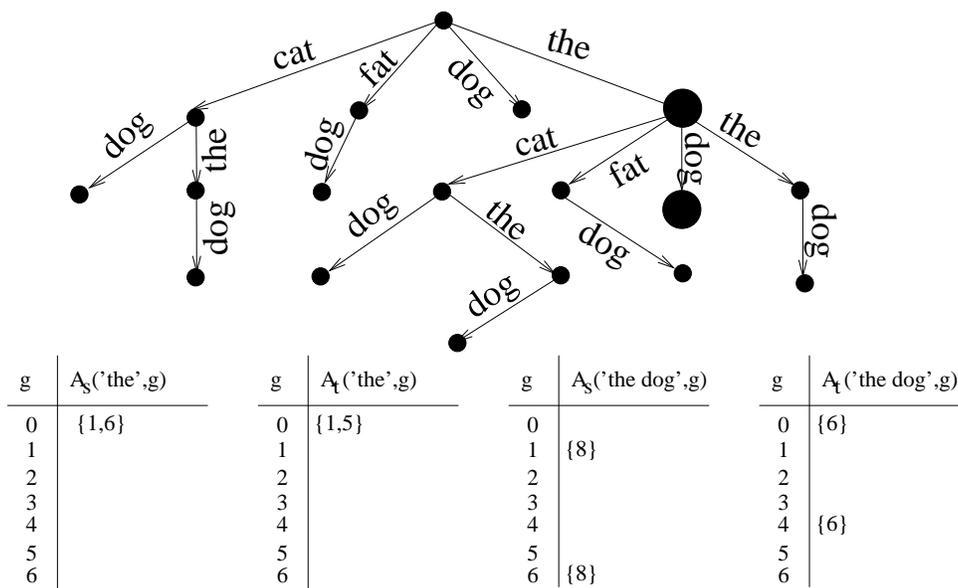


Figure 1: The co-occurrence trie of the example strings (top), and the sets of alive indices for the substrings 'the' and 'the dog' in the two strings with numbers of gaps ranging from 0 to 6 (bottom).

where the index set satisfies $U = \Sigma^p$ or $U = \Sigma^q \cup \dots \cup \Sigma^p$, and

$$\Phi_u(s) = 1/\lambda^p \sum_{i:u=s(i)} \lambda^{span(i)} \tag{4}$$

We will present three algorithms all of which are then experimentally compared in Section 4. The first is based on constructing an implicit trie data structure for the co-occurring gapped substrings in s and t . The second algorithm is the dynamic programming approach by Lodhi et al., (2002), and the third is a new method based on sparse dynamic programming.

As a running example we use two strings $s = 'The cat was chased by the fat dog'$ and $t = 'The fat cat bit the dog'$ and, for illustration, we apply the word alphabet. Hence, in the example, 'letters' corresponds to English words, 'substrings' and 'subsequences' to English phrases.

3.1 Trie-Based Computation

Trie-based computation (Leslie and Kuang, 2003; Cristianini and Shawe-Taylor, 2004) of the gap weighted subsequence kernel is based on making depth-first search into co-occurring subsequences in the two strings, starting from co-occurring one-letter matches and extending the matches letter by letter until the desired length p is reached. The search composes an implicit trie-structure of matching subsequences in the two strings: each path from root to a node corresponds to a subsequence that co-occurs in the two strings, in one or more locations, with number of gaps at most given integer g_{max} . The number of gaps need to be restricted in order to keep computation efficient.

Thus the resulting kernel can be considered as an approximation of (3) where co-occurrences with more than g_{max} gaps are discarded.

Figure 1 shows the trie structure of co-occurring subsequences for the example strings, when the index set is fixed to $U = \Sigma^3$, the three word phrases. Below we briefly describe a trie-based algorithm that is a slight variation of the one described by Shawe-Taylor and Cristianini (2004) and also bear resemblance to the related mismatch string kernel algorithm by Leslie et al. (2003).

In a trie-node corresponding to subsequence $u = u_1 \cdots u_q$ the algorithm keeps track of all matches $s(\mathbf{i}) = u$ that could still be extended further. For each such index set $\mathbf{i} = i_1 \dots i_q$, the last index i_q is stored in a list of *alive* matches $A_s(u, g)$ where $g = span(\mathbf{i}) - q$ is the number of gaps in the match. Similarly for t the lists $A_t(u, g)$ are maintained. To expand a node u , the algorithm looks for all possible letters the matching subsequence can be extended to a longer match uc . For an alive match $i \in A_s(u, g)$ and all $g' \leq g_{max} - g$ the algorithm puts the indices $i + 1 + g'$ into $A_s(us_{i+1+g'}, g + g')$. The lists $A_t(ut_{i+1+g'}, g + g')$ are constructed the same way. The search is continued for nodes uc for which both $(\bigcup_g A_s(uc, g))$ and $(\bigcup_g A_t(uc, g))$ are non-empty, that is, there is at least one occurrence of uc in both s and t , with some number of gaps. This makes the trie much sparser than the subsequence tries for either one of the strings alone would be.

In our example (Figure 1), 'the' is encountered in positions 1 and 6 of s , with (trivially) no gaps, and in positions 1 and 5 in t . To find the alive indices for 'the dog' the algorithm searches for the occurrence of 'dog', in s from indices 2 and 7 onwards, and finds the occurrence in position 6, corresponding to an occurrence with 1 and 4 gaps, respectively. Similar analysis is performed for t . When a node u in depth d is encountered one easily obtains the relevant terms in the kernel via computing the sum

$$\phi_u(s)\phi_u(t) = \sum_{g_s, g_t} \lambda^{g_s+d} |A_s(u, g_s)| \cdot \lambda^{g_t+d} |A_t(u, g_t)|.$$

If a length- p subsequence kernel is being computed this computation only need to be performed in the leaves of the trie. For bounded-length subsequence kernel, the computation needs to be performed in all trie nodes that are in depth $q \leq d \leq p$.

Note that the above approach differs from the (g, k) -gapped trie algorithm by Leslie and Kuang (2003) in two respects: First, the strings s and t are not broken into frames before the search but the algorithm maintains the lists $A_s(u, g)$ to keep track of the subsequence occurrences. Second, the algorithm keeps track of the number of gaps in the occurrences during the search. This relieves us from the need to embark on dynamic programming search in the trie leaves to compute the values $\phi_u(s)\phi_u(t)$.

The worst-case time complexity of the algorithm, $O(\binom{p+g_{max}}{p}(|s| + |t|))$, arises when $s = t$, which follows from noticing that each position in the two strings is a start location of a co-occurring subsequence, and there are $O(\binom{p+g_{max}}{p})$ possible combinations of assigning p letters and g_{max} gaps in a window of length $p + g_{max}$. Note that if no gaps are allowed we get the time complexity $O(p(|s| + |t|))$ matching the suffix tree approach.

3.2 Dynamic Programming Computation

The basis of dynamic programming computation of the string kernel (3) is the following observation: if there is a co-occurrence of substring $u_1 \dots u_q$ that ends in positions k 'th and l 'th position of s and t , respectively, two conditions must be satisfied:

1. there must be a matching pair of characters in the last positions: $s_k = t_l$, and

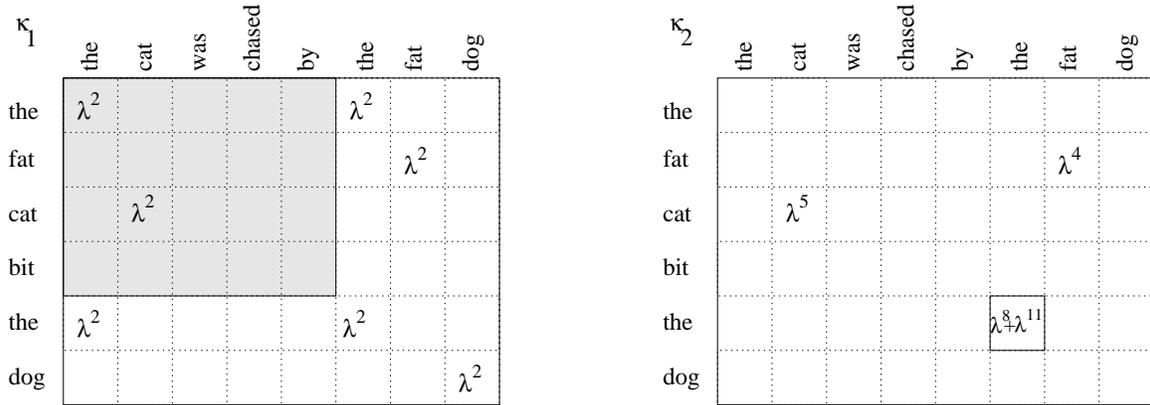


Figure 2: The co-occurrence weights of length-one (a) and length-two (b) substrings. Computing $\kappa_2(6,8)$ requires efficiently maintaining the appropriately scaled sum of the weights in the shaded region of κ_1 .

- there must be a matching prefix $u_1 \dots u_{q-1}$ that ends in some position $i < k$ in string s and some position $j < l$ in string t .

Moreover, ignoring the normalization $1/\lambda^{2q}$, the co-occurrence weight can be computed by from the weight of the prefix co-occurrence by extending the subsequences with $k - i$ and $l - j$ letters, respectively:

$$\lambda^{\text{span}([i_1, \dots, i_q, k])} \cdot \lambda^{\text{span}([j_1, \dots, j_q, l])} = \lambda^{\text{span}([i_1, \dots, i_q])} \cdot \lambda^{\text{span}([j_1, \dots, j_q])} \cdot \lambda^{k-i_q} \lambda^{l-j_q}.$$

Denoting by $\kappa_q(k, l)$ the sum of weights of length- q substring co-occurrences, again ignoring the normalization $1/\lambda^{2q}$, that end at positions k and l in s and t , respectively, the above observations can be summarized in the following recurrence

$$\kappa_q(k, l) = \begin{cases} \lambda^2 \mathbb{I}[s_k = t_l] & \text{for } q = 1, \text{ and} \\ \sum_{i < k} \sum_{j < l} \lambda^{k-i+l-j} \kappa_{q-1}(i, j) \mathbb{I}[s_k = t_l], & \text{for } q > 1, \end{cases} \quad (5)$$

where Figure 2 depicts the idea behind the recurrence (5): to compute $\kappa_2(5,6)$ we need to extend the length-1 matches in the shaded region, $\kappa_1(i, j), i < 5, j < 6$, into length-2 matches by adding gaps. The weights of two length-1 matches in positions (1,1) and (3,2) are rescaled before summation: $\lambda^8 + \lambda^{11} = \lambda^{5-3+6-2} \lambda^2 + \lambda^{5-1+6-1} \lambda^2$.

The dynamic programming algorithm (Figure 4) computes this recurrence by starting with subsequence length 1, which requires looping through all pairs of positions (i, j) in the two strings, checking for matching letters and summing up the co-occurrence weights. For convenience, the algorithm computes the sum of weights without the normalization $1/\lambda^{2q}$ that is applied in the final phase when computing the kernel value $K(q)$.

Longer subsequences are handled in increasing order of length, in accordance to (5). However, computing the double sum for each (k, l) (e.g. the shaded region in Figure 2) would be very inefficient, hence instead, a separate table storing the double sum

$$S(k, l) = \sum_{i < k} \sum_{j < l} \lambda^{k-i+l-j} \kappa_{q-1}(i, j)$$

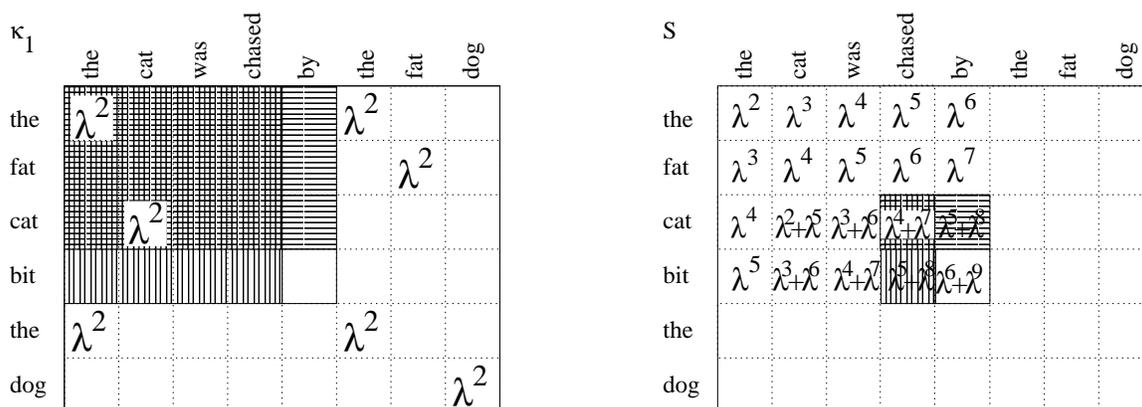


Figure 3: The table S (right) stores the scaled sums of co-occurrence weights of the prefixes of the strings. The value $S(k, l)$ is computed in constant time by adding up $\kappa_{q-1}(k, l) \lambda S(k-1, l)$ (horizontally striped area) and $\lambda S(k, l-1)$ (the vertically striped area) and subtracting $\lambda^2 S(k-1, l-1)$ (the intersection) that would otherwise be doubly counted.

is maintained. With that auxiliary table, computing the recurrence is very simple

$$\kappa_q(k, l) = \llbracket s_k = t_l \rrbracket \lambda^2 S(k-1, l-1).$$

Maintenance of the table S can be done efficiently via the relationship

$$S(k, l) = \kappa_{q-1}(k, l) + \lambda S(k-1, l) + \lambda S(k, l-1) - \lambda^2 S(k-1, l-1), \quad (6)$$

where the first term computes the contribution of the cell (k, l) , the two middle terms the regions $\{(i, j) | i < k, j \leq l\}$ and $\{(i, j) | i \leq k, j < l\}$, respectively, and the last term subtracts the twice counted region $\{(i, j) | i < k, j < l\}$.

In Figure 3 the computation of the value $S(4, 5) = \lambda^6 + \lambda^9$ is depicted. The value can be seen as the sum of weights of matching 'the ? ? ?' with 'the ? ? ?' (weight $\lambda^4 \cdot \lambda^5$) and 'the' ($\lambda^3 \cdot \lambda$), and 'cat' with 'cat ? ? ?' (weight $\lambda \cdot \lambda^5$).

The algorithm has time complexity $O(p|s||t|)$ which is immediately seen from the pseudocode in Figure 4. It is possible to optimize the algorithm to consume less memory. As the computation proceeds in increasing order of subsequence length and only the previous length is referred to, it suffices to maintain a single table κ that is reused for values $\kappa_1, \dots, \kappa_p$. Also, it suffices to maintain a one-dimensional vector $S(j)$ instead of the matrix $S(i, j)$, as the computation proceeds in the increasing order of i and only the values $S(i-1, :)$ are referred to when computing $S(i, :)$.

3.3 A Sparse Dynamic Programming Algorithm

In this section we describe a new algorithm for gap-weighted string kernel computation that is based on *sparse dynamic programming* (Eppstein et al., 1992). These algorithms utilise the fact that most entries in the dynamic programming matrix do not actually contribute to the results. The technique has been previously used, for example, to speed up transposition invariant string matching (Mäkinen, 2003) and, more close to our problem, in computing the longest-common subsequence of two strings given a fixed set of basis fragments (Baker and Giancarlo, 1998).

```

function K = DYNPROG(s,t,p,lambda)

 $\kappa_1 = \text{zeros}(\text{length}(s), \text{length}(t));$ 
K(1) = 0; % length-1 co-occurrences
for i = 1 : length(s)
    for j = 1 : length(t)
        if s(i) = t(j)
             $\kappa_1(i, j) = \lambda^2;$ 
            K(1) = K(1) +  $\kappa_1(i, j);$ 
        end
    end
end
K(1) = K(1)/ $\lambda^2;$  % renormalize

for l = 2 : p % co-occurrences of length 2...p
    K(l) = 0;
    S(1 : (l - 1), 1 : length(t)) = 0; S(l : length(s), 1 : (l - 1)) = 0;
    for i = 1 : length(s)
        for j = 1 : length(t)
            S(i, j) =  $\kappa_{l-1}(i, j) + \lambda \cdot S(i - 1, j) +$ 
                 $\lambda \cdot S(i, j - 1) - \lambda^2 \cdot S(i - 1, j - 1);$ 
            if s(i) = t(j)
                 $\kappa_l(i, j) = \lambda^2 \cdot S(i - 1, j - 1);$ 
                K(l) = K(l) +  $\kappa_l(i, j);$ 
            end
        end
    end
    K(l) = K(l)/ $\lambda^{2l};$  % renormalize
end

```

Figure 4: Dynamic programming algorithm for gap-weighted subsequence kernel computation. It takes a input two strings, subsequence length p and a penalty coefficient λ , and returns the kernel values $K(1), \dots, K(p)$ corresponding to different subsequence lengths.

Our algorithm is easiest to understand as a speed-up method for the dynamic programming approach described in Section 3.2. Despite its relatively low time-complexity, the algorithm makes unnecessary computations: the value $S(k, l)$ is required only when $s_k = t_l$, but computing that value using (6) dictates that all values $S(i, j), i \leq k, j \leq l$ are computed. In the following we present an algorithm that avoids these unnecessary computations via replacing the matrix S with a query tree that can be used to retrieve the values $S(i, j)$ as needed in $\log n$ time.

Another change to the original dynamic programming algorithm is that the matrix κ_q is replaced with a set of match lists

$$M_q(i) = ((j_1, \bar{\kappa}_q(i, j)), (j_2, \bar{\kappa}_q(i, j)), \dots)$$

where $\bar{\kappa}_q(i, j) = \lambda^{m-i+n-j} \cdot \kappa_q(i, j)$ can be interpreted as extending the length- q co-occurrences ending with s_i and t_j , respectively, with dummy gaps spanning positions $i + 1$ to m in s and positions $j + 1$ to n in t . The use of such dummy gap weighting relieves us from repeatedly scaling the kernel values as the search progresses: for any (k, l) it holds that

$$\bar{\kappa}_q(k, l) = \llbracket s_k = t_l \rrbracket \sum_{i < k} \sum_{j < l} \bar{\kappa}_{q-1}(i, j). \quad (7)$$

and the values $\bar{S}(k, l) = \sum_{i < k} \sum_{j < l} \bar{\kappa}_{q-1}(i, j)$ can be updated as the search proceeds without performing any rescaling of the items $\bar{S}(k, l) = \bar{S}(k - 1, l) + \sum_{j < l} \bar{\kappa}_{q-1}(k, j)$. This fact will be essential for maintaining our range-sum tree data structure, described below.

The data structure used for the queries belongs to the family of one-dimensional range query data structures, frequently used in computational geometry, online analytical processing (OLAP) and other fields where efficient range queries are needed (de Berg et al., 1997; Agarwal and Erickson, 1999).

The *range sum tree* for a set of key-value pairs $\mathcal{S} = \{(j_1, v_1), \dots, (j_h, v_h)\} \subset \{1, \dots, n\} \times \mathbb{R}$ is a binary tree of height $h = \lceil \log n \rceil$ where the nodes are in one-to-one correspondence with the keys in the range. The root contains the key 2^h , leaves contain all odd keys in the range and if an internal node in depth $d = \{0, \dots, h - 1\}$ contains key j , its left child contains the key $j - 2^{h-1-d}$ and the right child, when it exists, contains the key $j + 2^{h-1-d}$. With each key j in depth d a value is stored that represents a sum of item weights in a subrange $[j - 2^{h-d} + 1, j]$. It is easy to see that this subrange exactly covers the keys that are covered by the node's left subtree and the node itself. The range sum can be used to return the sum of values within an interval $[1, j]$ in $O(\log n)$ time by traversing the path from the node j to the root, and computing the sum

$$\text{Rangesum}([1, j]) = v_j + \sum_{\{h \in \text{Ancestors}(j) | h < j\}} v_h$$

Also, adding an item (j, v) to the tree takes time $O(\log n)$: we need to add v to node j and the set $\{h \in \text{Ancestors}(j) | h > j\}$.

For our algorithm, we will use the tree to query the values $\bar{S}(k, l) = \lambda^{m-k+n-l} S(k, l)$. To cope with this two-dimensional query region, we maintain the tree so that, when processing the match list $M_q(k)$ the tree will contain items $(j, v_j), 1 \leq j \leq n, v_j = \sum_{i < k} \bar{\kappa}_{q-1}(i, j)$, and thus the one-dimensional range query

$$\text{Rangesum}([1, l - 1]) = \sum_{j < l} v_j = \sum_{i < k} \sum_{j < l} \bar{\kappa}_{q-1}(i, j) = \bar{S}(k, l)$$

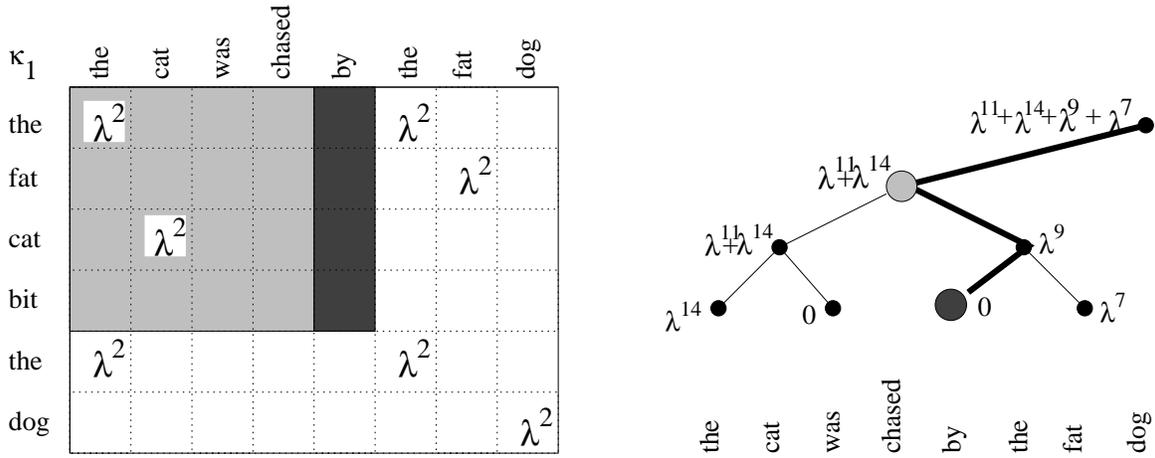


Figure 5: The range-sum tree on the right, with a query path (emphasized edges) and an area of the κ_1 matrix corresponding to the query path (grey and black strips) on the left. Starting from the leaf, the values in nodes that are left children of their parents are added together.

will return the desired answer.

Figure 5 depicts the state of the range-sum tree when computing the value $\bar{\kappa}_2(5, 6)$. The node 5 will contain value 0 as there are no subsequence co-occurrences within the strip $(i, 5), i < 5$. On the other hand, the key 4 will contain the value $\lambda^{11} + \lambda^{14}$ corresponding to the two co-occurrences within the (shaded) region $(i, j), i < 5, j \in [1, 4]$, scaled with the dummy gaps: $\lambda^{11} = \lambda^2 \lambda^{6-3+8-2}, \lambda^{14} = \lambda^2 \lambda^{6-1+8-1}$. The sum of weights in the shaded region is computed by adding the value 0 in node 5 corresponding to the empty black region, the value in node 4 corresponding to the grey region and skipping node 6 as $6 > 5$.

The sparse dynamic programming algorithm is shown in Figure 6. The algorithm takes as input a set $M_1 = \{M_1(1), \dots, M_1(m)\}$ of match lists

$$M_1(i) = ((j_1, \bar{\kappa}_1(i, j_1)), (j_2, \bar{\kappa}_1(i, j_2)), \dots),$$

that have been created in a preprocessing step taking $O(n + m + |\Sigma| + |M_1|)$ time and space. This preprocessing involves creating for each character $c \in \Sigma$ a list $I(c) = \{j | t_j = c\}$ of indices in the string t that contain the character c . To create a match list $M_1(i)$ then involves copying the indices in $I(s_i)$ to $M_1(i)$ and storing the corresponding values $\bar{\kappa}_1(i, j) = \lambda^{m-i+n-j} \lambda^2$ with the indices.

The main algorithm computes the kernel $K(s, t) = \kappa_p(m, n)$ by incrementally working out match sets M_2, \dots, M_p , corresponding to subsequence lengths $2, \dots, p$.

The processing of subsequence length q entails making one pass through the match sets $M_{q-1}(k)$ in increasing order of k . When constructing match list $M_q(k)$ the algorithm traverses match list $M_{q-1}(k)$, and for each item $(j, \bar{\kappa})$ in the list makes a range query $RangeSum([1, l-1]) = \sum_{i < k} \sum_{j < l} \bar{\kappa}_{q-1}(i, j)$, and, if the result is non-zero, inserts the item $(l, RangeSum([1, l-1]))$ to the list $M_q(k)$. After creating each list $M_q(k)$ the tree is updated with the contents of $M_{q-1}(k)$.

Finally, the kernel value $K(s, t)$ is computed by traversing the match lists $M_p(k), 1 \leq k \leq m$, rescaling the stored values to remove the dummy gap and summing up the rescaled values.

```

function  $K = \text{SPARSE}(M_1, p, \lambda)$ 
for  $q = 2 : (p - 1)$ 
     $\text{RangeSum}(1 : |t|) = 0$ ; % initially the ranges are empty
    for  $i = p : m$ 
        % compute the kappa values for the next level
         $M_q(i) = \{\}$ ;
        for  $(j_h, \bar{\kappa}_h) \in M_{q-1}(i)$ 
             $S = \text{query}(\text{RangeSum}, [1, j_h - 1])$ ; % make range query
            if  $S > 0$ 
                 $M_q(i) = M_q(i) \cup (j_h, S)$ ;
            end
        end
        % update the range with  $M_{q-1}(i)$ 
        for  $(j_h, \bar{\kappa}_h) \in M_{q-1}(i)$ 
             $\text{update}(\text{RangeSum}, (j_h, \bar{\kappa}_h))$ ;
        end
    end
end
    % compute the values for the final level
     $K = 0$ ;
    for  $i = p : (m - 1)$ 
        for  $(j_h, \kappa_h) \in M_{p-1}(i)$ ;
            if  $j_h < n$ 
                 $K = K + \bar{\kappa}_h \lambda^{i+j_h}$ ; % rescale to remove the dummy gap
            end
        end
    end
end

```

Figure 6: The algorithm for computing the gap-weighted subsequences kernel for two strings s and t . The algorithm takes as input a match set $M_1 = \{M_1(1), \dots, M_1(m)\}$, a penalty coefficient $0 < \lambda \leq 1$ and subsequence length p .

The queries and updates amount to $O(\log n)$ per item in the match list so the time complexity to process level q is $O(|M_q| \log n)$. Since we have $|M_1| \geq |M_2| \geq \dots \geq |M_p|$, the total time complexity will be $O(p|M_1| \log n)$. On random strings $|M_1| \approx |s||t|/|\Sigma|$. Hence, the sparse algorithm is likely to excel when $\log n/|\Sigma|$ is small, which we verify in the experiments presented in Section 4.

3.4 Variants and Implementation

The presented algorithm can be modified to compute many of the string kernel variants:

- A kernel only counting the number of co-occurrences of substrings is trivially obtained by setting $\lambda = 1$. In practical implementation, one can remove the scaling/rescaling operations from the algorithm, thus reducing the constants hidden in the asymptotic time-complexity.
- Bounded-length subsequence kernels are straight-forward to obtain. For each subsequence length $q \leq l \leq p$, the sum of weights in the match lists, rescaled to remove the dummy gaps, needs to be computed, as opposed for the length p only, as in the original algorithm. Thus, kernels of the form $K(s, t) = \sum_{q=1}^p w_q K_q(s, t)$, $w_q \in \mathbb{R}$ are easily obtained.
- Weighting by the number of gaps and the use of character specific gap penalties only require minor modifications to the algorithm (see below).

However, soft matching approaches (c.f Saunders et al., 2002), where most of the characters can be matched with each other with different costs (or utilities), are beyond this algorithm. This is because the efficiency of the algorithm relies on the match sets M_q to be sparse.

Weighting by the number of gaps. It is straightforward to modify the algorithm to penalize the number of gaps in the subsequence, instead of the total gap length. The kernel

$$\kappa_{Gap\#}(s, t) = \sum_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t), \text{ with } \phi_u^p(s) = \sum_{\mathbf{i}: u=s(\mathbf{i})} \lambda^{\sum_j [i_{j+1} - i_j > 1]}, u \in \Sigma^p,$$

can be computed via the recurrence

$$\begin{aligned} \kappa_q(k, l) = [s_i = t_j] & \left(\sum_{i < k-1, j < l-1} \lambda^2 \kappa_{q-1}(i, j) \right. \\ & \left. + \sum_{i < k-1} \lambda \kappa_{q-1}(i, l-1) + \sum_{j < l-1} \lambda \kappa_{q-1}(k-1, j) + \kappa_{q-1}(k-1, l-1) \right) \end{aligned} \quad (8)$$

which again leads to the $O(p|s||t|)$ time complexity. The first term takes into account co-occurrences where one gap is inserted to both the matched subsequences. The second and third term correspond to matches where a gap is inserted to occurrences in s only and t only, respectively. The last term takes into account matches where no gap is inserted to either s or t .

The sparse dynamic programming algorithm can be made to compute this recurrence efficiently by a simple modification: The range sum tree updates need to be lagged by one iteration so that, when creating the match list $M_q(k)$, the tree will not yet contain the values in the match list $M_{q-1}(k-1)$; these values will be added to the tree after handling the match list $M_q(k)$. By such an arrangement, the recurrence can be computed as

$$\kappa_q(k, l) = [s_i = t_j] \left(\lambda^2 \cdot \text{Rangesum}([1, l-2]) + \lambda \cdot \text{Rangesum}([l-1, l-1]) + \lambda r(j) + \kappa_{q-1}(k-1, l-1) \right),$$

where the values $r(j) = \sum_{j < l-1} \kappa_{q-1}(k-1, j)$ are incrementally computed while processing the match list $M_{q-1}(k-1)$. The algorithm's time complexity $O(p|M|\log n)$ will not change.

Character-specific gap and match weights. Another variant is to let the gap penalty depend on the character that was skipped, so that we have a set of penalties $\{\lambda_a | a \in \Sigma\}$. To implement this we need to precompute a vector $\Lambda_s = (\lambda_{s,k})_{k=1}^{|s|}$, $\lambda_{s,k} = \lambda_{s_1} \times \dots \times \lambda_{s_k}$ and an analogous vector Λ_t for string t . In the algorithm, when storing the item $\kappa_{q-1}(i, j)$ in the range sum tree, it is first scaled by $\lambda_{s,m}/\lambda_{s,i} \cdot \lambda_{t,n}/\lambda_{t,j}$ to introduce a dummy gaps $s(i+1 : m)$ and $t(j+1 : n)$ with character specific weighting. As with uniform gap weights, when computing the final level, rescaling by $\lambda_{s,k}/\lambda_{s,m} \cdot \lambda_{t,l}/\lambda_{t,n}$ is needed to recover the value $\kappa_p(k, l)$.

The approach can easily be extended to handle different weights for matches and gaps, as suggested by Cancedda et al., (2003). This only requires performing a scaling

$$\kappa_q(k, l) = \frac{\gamma_{s_k} \gamma_{t_l}}{\lambda_{s_k} \lambda_{t_l}} \text{RangeSum}([1, l-1]),$$

where γ_a and λ_a are the match and gap decays for letter a , respectively, to reflect the fact that s_k was matched to t_l rather than skipped over.

Implementation. The algorithm described above has been implemented in MATLAB 7.0. The code has been heavily tweaked to ensure that the benefits suggested by theoretical analysis can also be realized in practise. The major tweaks include

- **Range sum tree storage.** In our MATLAB implementation, the range sum tree is implicitly stored in a weight vector w storing the sum of the left subtree of each node $1 \leq j \leq n$. To speed up computations we also precompute in separate tables the nodes that need to be visited when querying or updating the range sum tree. For example, in the situation depicted in Figure (5) the precomputed *query path* will contain the nodes 4 and 5. The corresponding *update path* will contain the nodes 6 and 8.
- **Avoiding numerical underflow.** The algorithm in Figure 6 stores the items in the form $\lambda^{m-i+n-j} \kappa_{q-1}(i, j)$ and rescales them when computing the level p . This approach suffers from the potential of numerical underflow when handling long strings. In order to avoid that, we divide the index plane into rectangles of height and width sufficiently small (depending on the value λ) such that within a rectangle $[x', x''] \times [y', y'']$ the values are stored in the form $\lambda^{x''+y''-i-j}$. The handling of the boundaries of the rectangles causes a small additive overhead to the time complexity. The same technique can be used with the above discussed variants as well.

The implementation of the sparse dynamic programming algorithm is available via WWW from the home page of Juho Rousu: <http://www.cs.helsinki.fi/juho.rousu>.

4. Empirical Evaluation

We compared the time consumption of the following gapped string kernel algorithms, all implemented in MATLAB:

SPARSE. The new sparse dynamic programming algorithm.

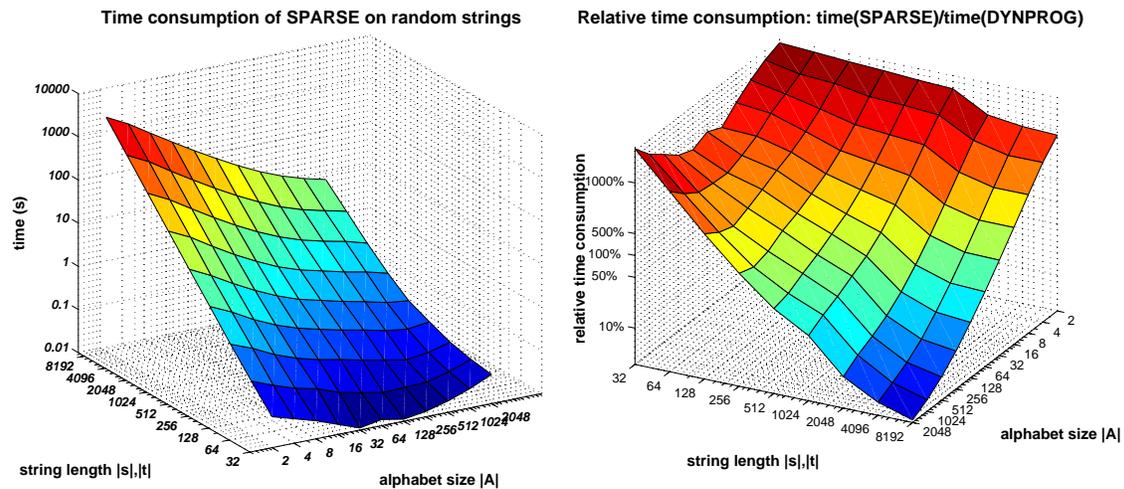


Figure 7: The time consumption of SPARSE in seconds (left) and relative time consumption of SPARSE relative to the time consumption of DYNPROG (right). The subsequence length $p = 10$ was used. Note the logarithmic scale on all axes.

DYNPROG. The full dynamic programming approach of Section 3.2.

TRIE. The Trie-based computation approach of Section 3.1.

Note that, differently from TRIE, DYNPROG and SPARSE place no hard restriction on the gap length but softly penalize the increase in gap length. We used three data sets for comparing the algorithms.

- Randomly generated strings, with varying length and alphabet sizes.
- 1000 random English news article pairs from the Reuters-21578 corpus, represented as sequences of syllables. The size of the syllable alphabet was 3769.
- 1000 random document pairs from the Chinese part of the Reuter's multilingual RCV-2 corpus. The size of the alphabet was 3142.

The tests were run on a 3GHZ Pentium 4 processor with 1.5GB main memory.

4.1 Results on Random Strings

In our first test we tested the time-consumption of the algorithm SPARSE as a function of string length and alphabet size. Figure 7 depicts the results. On the left the algorithms absolute time consumption is shown. The inverse dependency of the time-consumption on the alphabet size is clearly visible. Also, the larger the alphabet, the slower the time-consumption increases when the string length is increased.

On the right in Figure 7 the time-consumption of the sparse approach relative to the full dynamic programming approach is shown. With small alphabets and short strings DYNPROG is faster than

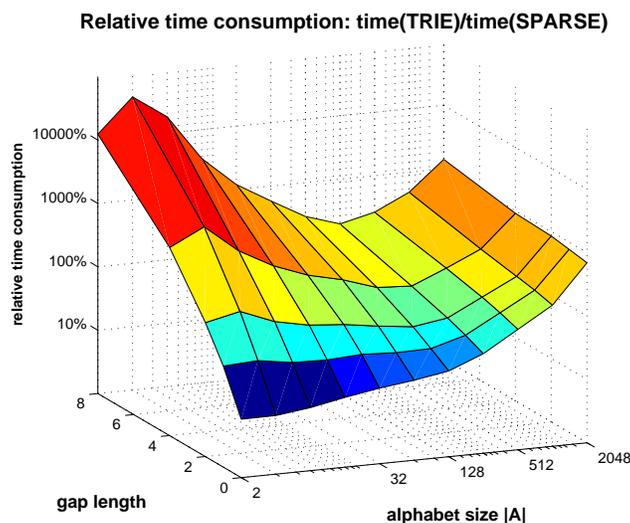


Figure 8: The relative time consumption $\text{time}(\text{TRIE})/\text{time}(\text{SPARSE})$ of the algorithms, as a function of alphabet size and gap length. The subsequence length $p = 10$ and string length $|s|, |t| = 512$ was used. Note the logarithmic scale.

SPARSE. With long strings and large alphabets the roles reverse: on strings longer than 1024 letters and alphabet sizes over 256, the SPARSE can be an order of magnitude faster than DYNPROG.

In our second test we compare the speed of TRIE algorithm to the SPARSE algorithm. Figure 8 depicts the relative time consumption as a function of alphabet size and gap length. Subsequence length of 10 and string length of 512 were used. Since SPARSE does not place any restriction to the gap length, in the comparison only the time taken by TRIE actually varied when the maximum number of gaps was varied.

The figure shows that TRIE algorithm gets very significantly slower than SPARSE when more gaps are allowed especially so on small alphabets. TRIE is faster than SPARSE only when the number of gaps is restricted to 2 or below. On very large alphabets even disallowing gaps does not bring TRIE below the time consumption of SPARSE.

The fastest algorithm as a function of string length and alphabet size is depicted in Figure 9, with different settings for the subsequence length (p) and the maximum number of gaps allowed in the TRIE algorithm. DYNPROG is the fastest method on short strings independent on the alphabet size and the subsequence length (a-d). If no gaps are allowed, TRIE is competitive on small to medium-size alphabets and long strings (a). When the subsequence length is increased, TRIE is faster than SPARSE even on large alphabets (c). The situation changes when gaps are allowed in TRIE algorithm: then SPARSE is the best method on large alphabets, and DYNPROG on small alphabets, TRIE excelling on medium-sized alphabets if long subsequences are searched for (d).

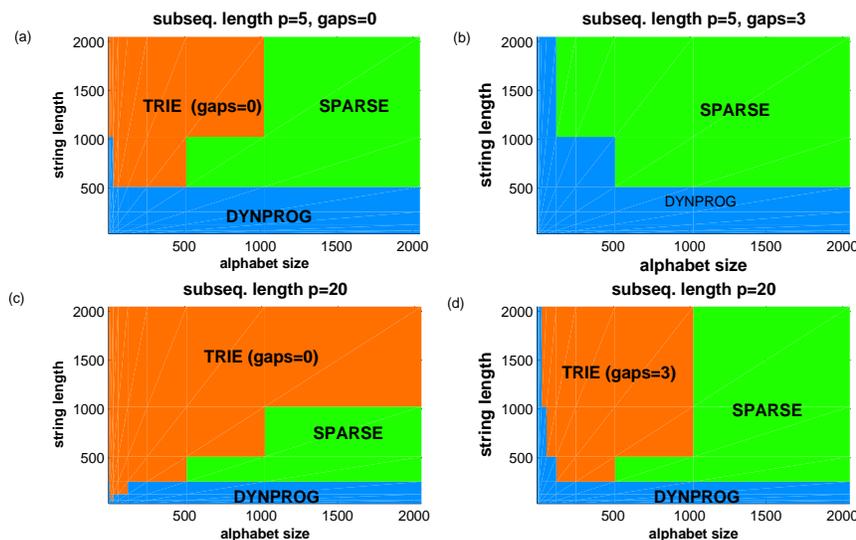


Figure 9: The fastest algorithm as a function of string length and alphabet size, with different subsequence lengths (p) and upper bounds for the number of gaps in the TRIE algorithm.

4.2 Results on Reuter’s News Articles

Our second set of experiments tests the speed of kernel computation on two Reuter’s newswire article data sets, English articles from Reuters-21578 corpus represented as sequences of syllables and Chinese articles from the multilingual RCV-2 corpus.

We computed the gap-weighted string kernel using DYNPROG and SPARSE for 1000 random document pairs, varying the subsequence lengths in the range 5-20. In our preliminary experiments, TRIE was significantly slower than both of the dynamic programming approaches on these data sets. Hence, we omitted that algorithm from the comparison.

The results on the English news articles are summarized in Figure 10. In the figure each document pair is plotted to the location corresponding to the (geometric) mean length of the documents (x-axis) and the inverse of the match frequency $|s||t|/|M|$ (y-axis), which also can be thought as the effective alphabet size: if the syllables were independently randomly drawn from alphabet of size $|\Sigma| = |s||t|/|M|$, the expected size of the match set would be $|M|$. Note that, due to the skewed distribution of syllables in the documents this number is usually significantly lower than the size of the syllable alphabet.

The marker type corresponds to the minimum subsequence length p required to make SPARSE run faster than DYNPROG on that document pair. Document pairs marked with ‘+’ require $p > 20$, circles require $11 \leq p \leq 20$, boxes require $6 \leq p \leq 10$, and for diamonds $p \leq 5$ is sufficient. Similar behaviour to that observed in the tests involving random strings can be seen: the longer the documents and the sparser the match matrix, the smaller value of p suffices to make SPARSE the faster algorithm.

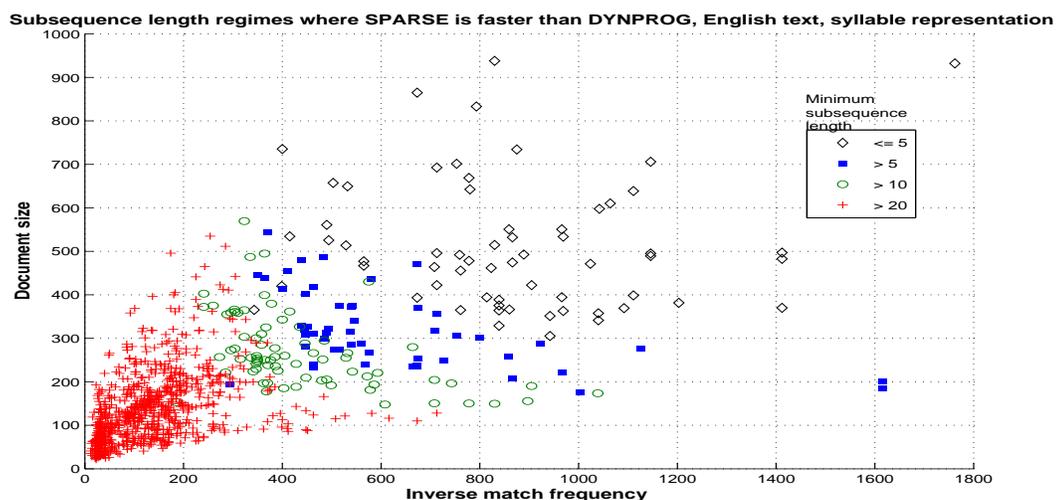


Figure 10: Regimes of subsequence length p , document size (x-axis) and inverse frequency of matching letters (y-axis), where SPARSE is faster than DYNPROG on English syllable-represented news articles. Document pairs marked with '+' require $p > 20$, circles require $11 \leq p \leq 20$, boxes require $6 \leq p \leq 10$, and for diamonds $p = 5$ is sufficient.

The results on Chinese news are summarized in Figure 11. The behaviour of the algorithms can be seen to be essentially the same as on the syllable converted English text: long documents and a sparse match matrix favour SPARSE.

5. Discussion and Open Problems

Based on the presented experiments, the full dynamic programming approach is the fastest method on short strings. On longer strings, the best algorithm depends on other parameters: if the alphabet is large the new sparse dynamic programming approach is the fastest method, if the alphabet is small DYNPROG is the best method. On medium-sized alphabets, the trie-based approach is competitive if the number of gaps can be strongly restricted.

The observed relative performance can be explained as follows. When the alphabet size is small, allowing more gaps rapidly expands the number of partially matching subsequences. Since TRIE explicitly keeps track of them, its time-consumption increases. SPARSE also suffers on small alphabets. However, it can never be worse than DYNPROG by more than a $\log n$ factor. On large alphabets, TRIE has an overhead of keeping track of all subtrees that may or may not need to be expanded. The improving performance of TRIE by increasing subsequence length is also easy to explain: the trie becomes the sparser the deeper the search level. Thus deepening the search is relatively cheap.

From the point of view algorithm development, an open question is whether the time-complexity of the sparse dynamic programming approach could be reduced below $O(p|M|\log n)$. The literature on geometric range searching does not offer a direct route forward: no index structures are known for one or two-dimensional range queries that can be maintained in less than amortized $O(\log n)$ time.

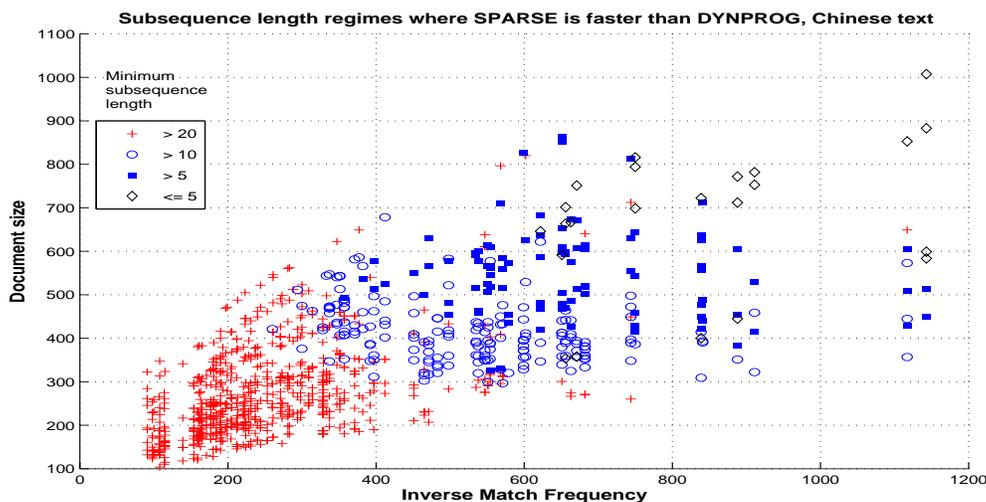


Figure 11: Regimes of subsequence length p , document size (x-axis) and inverse frequency of matching letters (y-axis), where SPARSE is faster than DYNPROG on Chinese news articles. Document pairs marked with '+' require $p > 20$, circles require $11 \leq p \leq 20$, boxes require $6 \leq p \leq 10$, and for diamonds $p = 5$ is sufficient.

per query (Agarwal and Erickson, 1999), even when taking advantage of the fact that the points are situated on an integer grid (Overmars, 1988, Alstrup et al., 2000). On the other hand the best lower bounds are of order $\Omega(\log \log n)$ per query (Chazelle, 1995).

There are amortized $O(\alpha(n))$ time range query methods, where $\alpha(n)$ is the inverse Ackermann's function, but they require $O(|s||t|)$ (Chazelle and Rosenberg, 1989) or $O(|M|^2)$ preprocessing (Poon, 2003)—which would lead to $O(p|s||t|)$ and $O(p|M|^2)$ total time complexity in our case, if applied in a straightforward manner. But, could we get $O(p|M|\alpha(|M|) + |s||t|)$ complexity by taking advantage of the fact that the match sets satisfy $M_1 \supset M_2 \supset \dots \supset M_p$? Moreover, each match set M_l is highly structured: for each $l \leq i \leq |s|$ we make the the range query *sequence* $[1, j_1] \subset [1, j_2] \subset \dots \subset [1, j_r]$, where $s_i = t_{j_h}, l < j_h \leq |t|$. In addition, for i and i' that satisfy $s_i = s_{i'}$ exactly the same query sequence is made. However, taking advantage of this appears to be non-trivial. For example, maintaining a separate range-query index for each character would result in $O(p|M||\Sigma|) \approx O(p|s||t|)$ time complexity.

6. Conclusions

We presented a sparse dynamic programming algorithm that efficiently computes the gap-weighted string kernels. The algorithm is easily adaptable to different string kernel variants, including fixed-length and bounded-length subsequence kernels and different gap penalization schemes, including penalization by total length of the gaps and number of the gaps as well as character specific gap/match penalization.

Our empirical results suggest that the sparse dynamic programming approach could be useful in text categorization applications when using syllable or word alphabets. Such alphabets have shown

to be useful in document classification tasks (Saunders et al., 2002, Cancedda et al., 2003). As the algorithm scales well to long documents when the alphabet is large, it could find use in classification of longer documents than the relatively short news stories, for instance, full-length research articles.

Acknowledgments

We thank Yaoyong Li and Craig Saunders for the help preparing the data sets. The work by Juho Rousu has been supported by the EU/Marie Curie Fellowship grant HPMF-CT-2002-02110. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

- P. Agarwal and J. Erickson. Geometric range searching and its relatives. *Contemporary Mathematics 23: Advances in Discrete and Computational Geometry*, pages 1–56, 1999.
- S. Alstrup, G.S. Brodal, T. Rauhe. New Data Structures for Orthogonal Range Searching. In *Proceedings of 41st Annual Symposium on Foundations of Computer Science*, pages 198–207, 2000.
- B. S. Baker and R. Giancarlo. Longest common subsequence from fragments via sparse dynamic programming. In *Proceedings of Sixth Annual European Symposium on Algorithms*, pages 79–90, 1998.
- M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry – Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
- N. Cancedda, E. Gaussier, C. Goutte, J-M. Renders. Word-Sequence Kernels. *Journal of Machine Learning Research* 3:1059–1082, 2003.
- B. Chazelle. Lower bounds for off-line range searching. In *Proceedings of 27th Annual ACM Symposium on Theory of Computing*, pages 733–740, 1995.
- B. Chazelle and B. Rosenberg. Computing partial sums in multidimensional arrays. In *Proceedings of 5th Annual Symposium on Computational Geometry*, pages 131–139, 1989.
- N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines*. Cambridge University Press, 2000.
- D. Eppstein. Efficient algorithms for sequence analysis with concave and convex gap costs. Ph.D. thesis, Columbia University, 1989.
- D. Eppstein, Z. Galil, R. Giancarlo, and G. F. Italiano. Sparse dynamic programming I: linear cost functions. *Journal of the ACM* 39, 3:519–545, 1992.
- D. Haussler. Convolution kernels on discrete structures. Technical report, University of California, Santa Cruz, 1999.
- T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of Tenth European Conference on Machine Learning*, pages 137–142, 1999.
- C. Leslie, E. Eskin and W. Stafford Noble. The spectrum kernel: a string kernel for SVM protein classification. In *Proceedings of Pacific Symposium on Biocomputing* 7, pages 566–575, 2002.
- C. Leslie, E. Eskin, J. Weston and W. Stafford Noble. Mismatch String Kernels for SVM Protein Classification. *Advances in Neural Information Processing Systems* 15, pages 1417–1424, 2003.

- C. Leslie and R. Kuang. Fast Kernels for Inexact String Matching. In *Proceedings of 16th Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel'2003. Lecture Notes in Computer Science 2777*:114 - 128, 2003.
- H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini and C. Watkins. Text Classification using String Kernels. *Journal of Machine Learning Research* 2:419–444, 2002.
- V. Mäkinen. Parameterized Approximate String Matching and Local-Similarity-Based Point-Pattern Matching. Report A-2003-6. Department of Computer Science, University of Helsinki, 2003.
- M. Overmars. Efficient data structures for range searching on a grid. *Journal of Algorithms* 9:254–275, 1988.
- C. Poon. Dynamic orthogonal range queries in OLAP. *Theoretical Computer Science* 296(3):487–510, 2003.
- G. Salton, A. Wong and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM* 18(11):613–620, 1973.
- C. Saunders, H. Tschach and J. Shawe-Taylor. Syllables and other String Kernel Extensions. In *Proceedings of 19th International Conference on Machine Learning*, pages 530–537, 2002.
- J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.
- S. Vishwanathan and A. Smola. Fast Kernels for String and Tree Matching. *Advances in Neural Information Processing Systems 15*, pages 569–576, 2003,
- C. Watkins. Dynamic alignment kernels. In A.J. Smola, P. Bartlett, B. Schölkopf and D. Schuurmans, eds., *Advances in Large Margin Classifiers*, pages 39–50, 2000,

Clustering on the Unit Hypersphere using von Mises-Fisher Distributions

Arindam Banerjee

*Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712, USA*

ABANERJE@ECE.UTEXAS.EDU

Inderjit S. Dhillon

*Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA*

INDERJIT@CS.UTEXAS.EDU

Joydeep Ghosh

*Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712, USA*

GHOSH@ECE.UTEXAS.EDU

Suvrit Sra

*Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA*

SUVRIT@CS.UTEXAS.EDU

Editor: Greg Ridgeway

Abstract

Several large scale data mining applications, such as text categorization and gene expression analysis, involve high-dimensional data that is also inherently directional in nature. Often such data is L_2 normalized so that it lies on the surface of a unit hypersphere. Popular models such as (mixtures of) multi-variate Gaussians are inadequate for characterizing such data. This paper proposes a generative mixture-model approach to clustering directional data based on the von Mises-Fisher (vMF) distribution, which arises naturally for data distributed on the unit hypersphere. In particular, we derive and analyze two variants of the Expectation Maximization (EM) framework for estimating the mean and concentration parameters of this mixture. Numerical estimation of the concentration parameters is non-trivial in high dimensions since it involves functional inversion of ratios of Bessel functions. We also formulate two clustering algorithms corresponding to the variants of EM that we derive. Our approach provides a theoretical basis for the use of cosine similarity that has been widely employed by the information retrieval community, and obtains the spherical kmeans algorithm (kmeans with cosine similarity) as a special case of both variants. Empirical results on clustering of high-dimensional text and gene-expression data based on a mixture of vMF distributions show that the ability to estimate the concentration parameter for each vMF component, which is not present in existing approaches, yields superior results, especially for difficult clustering tasks in high-dimensional spaces.

Keywords: clustering, directional distributions, mixtures, von Mises-Fisher, expectation maximization, maximum likelihood, high dimensional data

1. Introduction

Clustering or segmentation of data is a fundamental data analysis step that has been actively investigated by many research communities over the past few decades (Jain and Dubes, 1988). However, traditional methods for clustering data are severely challenged by a variety of complex characteristics exhibited by certain recent data sets examined by the machine learning and data mining communities. These data sets, acquired from scientific domains and the world wide web, also impose significant demands on scalability, visualization and evaluation of clustering methods (Ghosh, 2003). In this paper we focus on clustering objects such as text documents and gene expressions, where the complexity arises from their representation as vectors that are not only very high dimensional (and often sparse) but also *directional*, i.e., the vector direction is relevant, not its magnitude.

One can broadly categorize clustering approaches to be either generative (also known as parametric or probabilistic) (Smyth, 1997; Jaakkola and Haussler, 1999) or discriminative (non-parametric) (Indyk, 1999; Schölkopf and Smola, 2001). The performance of an approach, and of a specific method within that approach, is quite data dependent; there is no clustering method that works the best across all types of data. Generative models, however, often provide greater insight into the anatomy of the clusters. A lot of domain knowledge can be incorporated into generative models, so that clustering of data uncovers specific desirable patterns that one is looking for.

Clustering algorithms using the generative model framework, often involve an appropriate application of the Expectation Maximization (EM) algorithm (Dempster et al., 1977; McLachlan and Krishnan, 1997) on a properly chosen statistical generative model for the data under consideration. For vector data, there are well studied clustering algorithms for popular generative models such as a mixture of multivariate Gaussians, whose effect is analogous to the use of Euclidean or Mahalanobis type distances as the chosen measure of distortion from the discriminative perspective.

The choice of a particular distortion measure (or the corresponding generative model) can be crucial to the performance of a clustering procedure. There are several domains where methods based on minimizing Euclidean distortions yield poor results (Strehl et al., 2000). For example, studies in information retrieval applications convincingly demonstrate *cosine similarity* to be a more effective measure of similarity for analyzing and clustering text documents. In this domain, there is substantial empirical evidence that normalizing the data vectors helps to remove the biases induced by the length of a document and provide superior results (Salton and McGill, 1983; Salton and Buckley, 1988). Further, the spherical kmeans (spkmeans) algorithm (Dhillon and Modha, 2001), that performs kmeans using cosine similarity instead of Euclidean distortion, has been found to work well for text clustering. Data Sets from such domains, where similarity measures such as cosine, Jaccard or Dice (Rasmussen, 1992) are more effective than measures derived from Mahalanobis type distances, possess intrinsic “directional” characteristics, and are hence better modeled as *directional data* (Mardia and Jupp, 2000).¹

There are many other important domains such as bioinformatics (e.g., Eisen et al. (1998)), collaborative filtering (e.g., Sarwar et al. (2001)) etc., in which directional data is encountered. Consider the Pearson correlation coefficient, which is a popular similarity measure in both these domains. Given $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the Pearson product moment correlation between them is given by $\rho(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^d (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^d (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^d (y_i - \bar{y})^2}}$, where $\bar{x} = \frac{1}{d} \sum_{i=1}^d x_i$, $\bar{y} = \frac{1}{d} \sum_{i=1}^d y_i$. Consider the mapping $\mathbf{x} \mapsto \tilde{\mathbf{x}}$ such that $\tilde{x}_i = \frac{x_i - \bar{x}}{\sqrt{\sum_{i=1}^d (x_i - \bar{x})^2}}$, and a similar mapping for \mathbf{y} . Then we have $\rho(\mathbf{x}, \mathbf{y}) = \tilde{\mathbf{x}}^T \tilde{\mathbf{y}}$. Moreover,

1. This paper treats L_2 normalized data and directional data as synonymous.

$\|\tilde{\mathbf{x}}\|_2 = \|\tilde{\mathbf{y}}\|_2 = 1$. Thus, the Pearson correlation is exactly the cosine similarity between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$. Hence, analysis and clustering of data using Pearson correlations is essentially a clustering problem for directional data.

1.1 Contributions

In this paper² we present a generative model, consisting of a mixture of von Mises-Fisher (vMF) distributions, tailored for directional data distributed on the surface of a unit hypersphere. We derive two clustering algorithms based on EM for estimating the parameters of the mixture model from first principles. The algorithm involves estimating a *concentration* parameter, κ , for high dimensional data. The ability to adapt κ on a per-component basis leads to substantial performance improvements over existing generative approaches to modeling directional data. We show a connection between the proposed methods and a class of existing algorithms for clustering high-dimensional directional data. In particular, our generative model has the same relation to `spkmeans` as a model based on a mixture of unit covariance Gaussians has to classical `kmeans` that uses squared Euclidean distances. We also present detailed experimental comparisons of the proposed algorithms with `spkmeans` and one of its variants. Our formulation uncovers the theoretical justification behind the use of the cosine similarity measure that has largely been ad-hoc, i.e., based on empirical or intuitive justification, so far.

Other key contributions of the paper are:

- It exposes the vMF model to the learning community and presents a detailed parameter estimation method for learning mixtures of vMF distributions in high-dimensions. Previously known parameter estimates for vMF distributions are reasonable only for low-dimensional data (typically only 2 or 3 dimensional data is considered) and are hence not applicable to many modern applications such as text clustering.
- We show that hard assignments maximize a tight lower bound on the incomplete log-likelihood function. In addition, our analysis of hard assignments is applicable to any mixture model learning using EM. This result is particularly important when using mixtures of vMFs since the computational needs for hard assignments are lower than what is required for the standard soft assignments (E-step) for these models.
- Extensive experimental results are provided on benchmark text and gene-expression data sets to show the efficacy of the proposed algorithms for high-dimensional, directional data. Good results are obtained even for fairly skewed data sets. A recent study (Banerjee and Langford, 2004) using PAC-MDL bounds for evaluation of clustering algorithms also demonstrated the efficacy of the proposed approaches.
- An explanation of the superior performance of the soft-assignment algorithm is obtained by drawing an analogy between the observed cluster formation behavior and locally adaptive annealing. See Section 7 for further details.

2. An earlier, short version of this paper appeared as: *Generative Model-based Clustering of Directional Data* in Proceedings KDD, 2003.

1.2 Related Work

There has been an enormous amount of work on clustering a wide variety of data sets across multiple disciplines over the past fifty years (Jain and Dubes, 1988). The methods presented in this paper are tailored for high-dimensional data with directional characteristics, rather than for arbitrary data sets. In the learning community, perhaps the most widely studied high-dimensional directional data stem from text documents represented by vector space models. Much of the work in this domain uses discriminative approaches (Steinbach et al., 2000; Zhao and Karypis, 2004). For example, hierarchical agglomerative methods based on cosine, Jaccard or Dice coefficients were dominant for text clustering till the mid-1990s (Rasmussen, 1992). Over the past few years several new approaches, ranging from spectral partitioning (Kannan et al., 2000; Zhao and Karypis, 2004), to the use of generative models from the exponential family, e.g., mixture of multinomials or Bernoulli distributions (Nigam et al., 2000) etc., have emerged. A fairly extensive list of references on generative approaches to text clustering can be found in (Zhong and Ghosh, 2003a).

Of particular relevance to this work is the `spkmeans` algorithm (Dhillon and Modha, 2001), which adapts the `kmeans` algorithm to normalized data by using the cosine similarity for cluster allocation, and also by re-normalizing the cluster means to unit length. The `spkmeans` algorithm is superior to regular `kmeans` for high-dimensional text data, and competitive or superior in both performance and speed to a wide range of other existing alternatives for text clustering (Strehl et al., 2000). It also provides better characterization of clusters in terms of their top representative or discriminative terms.

The larger topic of clustering very high-dimensional data (dimension in the thousands or more), irrespective of whether it is directional or not, has also attracted great interest lately. Again, most of the proposed methods of dealing with the curse of dimensionality in this context follow a density-based heuristic or a discriminatory approach (Ghosh, 2003). Among generative approaches for clustering high-dimensional data, perhaps the most noteworthy is one that uses low dimensional projections of mixtures of Gaussians (Dasgupta, 1999). It turns out that one of our proposed methods alleviates problems associated with high dimensionality via an implicit local annealing behavior.

The `vMF` distribution is known in the literature on directional statistics (Mardia and Jupp, 2000), and the maximum likelihood estimates (MLE) of the parameters have been given for a single distribution. Recently Piater (2001) obtained parameter estimates for a mixture for circular, i.e., 2-dimensional `vMFs`. In an Appendix to his thesis, Piater (2001) starts on an EM formulation for 2-D `vMFs` but cites the difficulty of parameter estimation (especially κ) and eventually avoids doing EM in favor of another numerical gradient descent based scheme. Mooney et al. (2003) use a mixture of two circular von Mises distributions to estimate the parameters using a quasi-Newton procedure. Wallace and Dowe (2000) perform mixture modeling for circular von Mises distributions and have produced a software called `Snob` that implements their ideas. McLachlan and Peel (2000) discuss mixture analysis of directional data and mention the possibility of using Fisher distributions (3-dimensional `vMFs`), but instead use 3-dimensional Kent distributions (Mardia and Jupp, 2000). They also mention work related to the clustering of directional data, but all the efforts included by them are restricted to 2-D or 3-D `vMFs`. Indeed, McLachlan and Peel (2000) also draw attention to the difficulty of parameter estimation even for 3-D `vMFs`. Even for a single component, the maximum-likelihood estimate for the concentration parameter κ involves inverting a ratio of two Bessel functions, and current ways of approximating this operation are inadequate for high-

dimensional data. It turns out that our estimate for κ translates into a substantial improvement in the empirical results.

The connection between a generative model involving vMF distributions with constant κ and the `spkmeans` algorithm was first observed by Banerjee and Ghosh (2002). A variant that could adapt in an on-line fashion leading to balanced clustering solutions was developed by Banerjee and Ghosh (2004). Balancing was encouraged by taking a frequency-sensitive competitive learning approach in which the concentration of a mixture component was made inversely proportional to the number of data points already allocated to it. Another online competitive learning scheme using vMF distributions for minimizing a KL-divergence based distortion was proposed by Sinkkonen and Kaski (2001). Note that the full EM solution was not obtained or employed in either of these works. Recently a detailed empirical study of several generative models for document clustering, including a simple mixture-of-vMFs model that constrains the concentration κ to be the same for all mixture components during any iteration was presented by Zhong and Ghosh (2003b). Even with this restriction, this model was superior to both hard and soft versions of multivariate Bernoulli and multinomial models. These positive results further motivate the current paper in which we present the general EM solution for parameter estimation of a mixture of vMF distributions. This enhancement leads to even better clustering performance for difficult clustering tasks: when clusters overlap, when cluster sizes are skewed, and when cluster sizes are small relative to the dimensionality of the data. In the process, several new, key insights into the nature of hard vs. soft mixture modeling and the behavior of vMF based mixture models are obtained.

The remainder of the paper is organized as follows. We review the multi-variate vMF distribution in Section 2. In Section 3 we introduce a generative model using a mixture of vMF distributions. We then derive the maximum likelihood parameter estimates of this model by employing an EM framework. Section 4 highlights our new method of approximating κ and also presents a mathematical analysis of hard assignments. Sections 3 and 4 form the basis for two clustering algorithms using soft and hard-assignments respectively, that are proposed in Section 5. Detailed experimental results and comparisons with other algorithms are offered in Section 6. A discussion on the behavior of our algorithms and a connection with simulated annealing follows in Section 7. Section 8 concludes our paper and highlights some possible directions for future work.

Notation. Bold faced variables, e.g., \mathbf{x} , $\boldsymbol{\mu}$ represent vectors; the norm $\|\cdot\|$ denotes the L_2 norm; sets are represented by script-style upper-case letters, e.g., \mathcal{X} , \mathcal{Z} . The set of reals is denoted by \mathbb{R} , while \mathbb{S}^{d-1} denotes the $(d-1)$ -dimensional sphere embedded in \mathbb{R}^d . Probability density functions are denoted by lower case letters such as f , p , q and the probability of a set of events is denoted by P . If a random vector \mathbf{x} is distributed as $p(\cdot)$, expectations of functions of \mathbf{x} are denoted by $E_p[\cdot]$.

2. Preliminaries

In this section, we review the von Mises-Fisher distribution and maximum likelihood estimation of its parameters from independent samples.

2.1 The von Mises-Fisher (vMF) Distribution

A d -dimensional unit random vector \mathbf{x} (i.e., $\mathbf{x} \in \mathbb{R}^d$ and $\|\mathbf{x}\| = 1$, or equivalently $\mathbf{x} \in \mathbb{S}^{d-1}$) is said to have d -variate von Mises-Fisher (vMF) distribution if its probability density function is given by

$$f(\mathbf{x}|\boldsymbol{\mu}, \kappa) = c_d(\kappa)e^{\kappa\boldsymbol{\mu}^T\mathbf{x}}, \quad (2.1)$$

where $\|\boldsymbol{\mu}\| = 1$, $\kappa \geq 0$ and $d \geq 2$. The normalizing constant $c_d(\kappa)$ is given by

$$c_d(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} I_{d/2-1}(\kappa)}, \tag{2.2}$$

where $I_r(\cdot)$ represents the modified Bessel function of the first kind and order r . The density $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ is parameterized by the mean direction $\boldsymbol{\mu}$, and the *concentration* parameter κ , so-called because it characterizes how strongly the unit vectors drawn according to $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ are concentrated about the mean direction $\boldsymbol{\mu}$. Larger values of κ imply stronger concentration about the mean direction. In particular when $\kappa = 0$, $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ reduces to the uniform density on \mathbb{S}^{d-1} , and as $\kappa \rightarrow \infty$, $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ tends to a point density. The interested reader is referred to Mardia and Jupp (2000), Fisher (1996) or Dhillon and Sra (2003) for details on vMF distributions.

The vMF distribution is one of the simplest parametric distributions for directional data, and has properties analogous to those of the multi-variate Gaussian distribution for data in \mathbb{R}^d . For example, the maximum entropy density on \mathbb{S}^{d-1} subject to the constraint that $E[\mathbf{x}]$ is fixed is a vMF density (see Rao (1973, pp. 172–174) and Mardia (1975) for details).

2.2 Maximum Likelihood Estimates

In this section we look briefly at maximum likelihood estimates for the parameters of a single vMF distribution. The detailed derivation can be found in Appendix A. Let \mathcal{X} be a finite set of sample unit vectors drawn independently following $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ (2.1), i.e.,

$$\mathcal{X} = \{\mathbf{x}_i \in \mathbb{S}^{d-1} \mid \mathbf{x}_i \text{ drawn following } f(\mathbf{x}|\boldsymbol{\mu}, \kappa) \text{ for } 1 \leq i \leq n\}.$$

Given \mathcal{X} we want to find maximum likelihood estimates for the parameters $\boldsymbol{\mu}$ and κ of the distribution $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$. Assuming the \mathbf{x}_i to be independent and identically distributed, we can write the log-likelihood of \mathcal{X} as

$$\ln P(\mathcal{X}|\boldsymbol{\mu}, \kappa) = n \ln c_d(\kappa) + \kappa \boldsymbol{\mu}^T \mathbf{r}, \tag{2.3}$$

where $\mathbf{r} = \sum_i \mathbf{x}_i$. To obtain the maximum likelihood estimates of $\boldsymbol{\mu}$ and κ , we have to maximize (2.3) subject to the constraints $\boldsymbol{\mu}^T \boldsymbol{\mu} = 1$ and $\kappa \geq 0$. After some algebra (details may be found in Section A.1) we find that the MLE solutions $\hat{\boldsymbol{\mu}}$ and $\hat{\kappa}$ may be obtained from the following equations:

$$\hat{\boldsymbol{\mu}} = \frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{\sum_{i=1}^n \mathbf{x}_i}{\|\sum_{i=1}^n \mathbf{x}_i\|}, \tag{2.4}$$

$$\text{and } \frac{I_{d/2}(\hat{\kappa})}{I_{d/2-1}(\hat{\kappa})} = \frac{\|\mathbf{r}\|}{n} = \bar{r}. \tag{2.5}$$

Since computing $\hat{\kappa}$ involves an implicit equation (2.5) that is a ratio of Bessel functions, it is not possible to obtain an analytic solution, and we have to take recourse to numerical or asymptotic methods to obtain an approximation (see Section 4.1).

3. EM on a Mixture of vMFs (moVMF)

We now consider a mixture of k vMF (moVMF) distributions that serves as a generative model for directional data. Subsequently we derive the update equations for estimating the mixture-density parameters from a given data set using the Expectation Maximization (EM) framework. Let $f_h(\mathbf{x}|\boldsymbol{\theta}_h)$

denote a vMF distribution with parameter $\theta_h = (\boldsymbol{\mu}_h, \kappa_h)$ for $1 \leq h \leq k$. Then a mixture of these k vMF distributions has a density given by

$$f(\mathbf{x}|\Theta) = \sum_{h=1}^k \alpha_h f_h(\mathbf{x}|\theta_h), \quad (3.1)$$

where $\Theta = \{\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k\}$ and the α_h are non-negative and sum to one. To sample a point from this mixture density we choose the h -th vMF randomly with probability α_h , and then sample a point (on \mathbb{S}^{d-1}) following $f_h(\mathbf{x}|\theta_h)$. Let $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be a data set of n independently sampled points that follow (3.1). Let $\mathcal{Z} = \{z_1, \dots, z_n\}$ be the corresponding set of hidden random variables that indicate the particular vMF distribution from which the points are sampled. In particular, $z_i = h$ if \mathbf{x}_i is sampled from $f_h(\mathbf{x}|\theta_h)$. Assuming that the values in the set \mathcal{Z} are known, the log-likelihood of the observed data is given by

$$\ln P(\mathcal{X}, \mathcal{Z}|\Theta) = \sum_{i=1}^n \ln(\alpha_{z_i} f_{z_i}(\mathbf{x}_i|\theta_{z_i})). \quad (3.2)$$

Obtaining maximum likelihood estimates for the parameters would have been easy were the z_i truly known. Unfortunately that is not the case, and (3.2) is really a random variable dependent on the distribution of \mathcal{Z} —this random variable is usually called the *complete data log-likelihood*. For a given (\mathcal{X}, Θ) , it is possible to estimate the most likely conditional distribution of $\mathcal{Z}|\mathcal{X}, \Theta$, and this estimation forms the E-step in an EM framework.

Using an EM approach for maximizing the expectation of (3.2) with the constraints $\boldsymbol{\mu}_h^T \boldsymbol{\mu}_h = 1$ and $\kappa_h \geq 0$, we obtain (see Appendix A.2),

$$\alpha_h = \frac{1}{n} \sum_{i=1}^n p(h|\mathbf{x}_i, \Theta), \quad (3.3)$$

$$\mathbf{r}_h = \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta), \quad (3.4)$$

$$\hat{\boldsymbol{\mu}}_h = \frac{\mathbf{r}_h}{\|\mathbf{r}_h\|}, \quad (3.5)$$

$$\frac{I_{d/2}(\hat{\kappa}_h)}{I_{d/2-1}(\hat{\kappa}_h)} = \frac{\|\mathbf{r}_h\|}{\sum_{i=1}^n p(h|\mathbf{x}_i, \Theta)}. \quad (3.6)$$

Observe that (3.5) and (3.6) are intuitive generalizations of (2.4) and (2.5) respectively, and they correspond to an M-step in an EM framework. Given these parameter updates, we now look at schemes for updating the distributions of $\mathcal{Z}|\mathcal{X}, \Theta$ (i.e., an E-step) to maximize the likelihood of the data given the parameters estimates above.

From the standard EM framework, the distribution of the hidden variables (Neal and Hinton, 1998; Bilmes, 1997) is given by

$$p(h|\mathbf{x}_i, \Theta) = \frac{\alpha_h f_h(\mathbf{x}_i|\Theta)}{\sum_{l=1}^k \alpha_l f_l(\mathbf{x}_i|\Theta)}. \quad (3.7)$$

It can be shown (Collins, 1997) that the *incomplete data log-likelihood*, $\ln p(\mathcal{X}|\Theta)$, is non-decreasing at each iteration of the parameter and distribution updates. Iteration over these two updates provides the foundation for our `soft-moVMF` algorithm given in Section 5.

Our second update scheme is based on the widely used hard-assignment heuristic for unsupervised learning. In this case, the distribution of the hidden variables is given by

$$q(h|\mathbf{x}_i, \Theta) = \begin{cases} 1, & \text{if } h = \operatorname{argmax}_{h'} p(h'|\mathbf{x}_i, \Theta), \\ 0, & \text{otherwise.} \end{cases} \quad (3.8)$$

We analyze the above hard-assignment rule in Section 4, and show that it maximizes a lower bound on the incomplete data log-likelihood. Iteration over the M-step and the hard-assignment rule leads to the `hard-moVMF` algorithm given in Section 5.

4. Handling Large and High-Dimensional Data Sets

Although the mixture model outlined in section 3 seems quite straight-forward, there are some of critical issues that need to be addressed before one can apply the model to large high-dimensional data sets:

- A. How to compute $\kappa_h, h = 1, \dots, k$ from (3.6) for high-dimensional data?
- B. Is it possible to significantly reduce computations and still get a reasonable clustering?

We address both these issues in this section, as they are significant for large high-dimensional data sets. The problem of estimating κ_h is analyzed in Section 4.1. In Section 4.2 we show that hard assignments can reduce computations significantly while giving a reasonable clustering.

4.1 Approximating κ

Recall that because of the lack of an analytical solution, it is not possible to directly estimate the κ values (see (2.5) and (3.6)). One may employ a nonlinear root-finder for estimating κ , but for high dimensional data, problems of overflows and numerical instabilities plague such root-finders. Therefore, an asymptotic approximation of κ is the best choice for estimating κ . Such approaches also have the benefit of taking constant computation time as opposed to any iterative method.

Mardia and Jupp (2000) provided approximations for estimating κ for a single component (2.5), for two limiting cases (Approximations (10.3.7) and (10.3.10) of Mardia and Jupp (2000, pp. 198)):

$$\hat{\kappa} \approx \frac{d-1}{2(1-\bar{r})} \quad \text{valid for large } \bar{r}, \quad (4.1)$$

$$\hat{\kappa} \approx d\bar{r} \left(1 + \frac{d}{d+2}\bar{r}^2 + \frac{d^2(d+8)}{(d+2)^2(d+4)}\bar{r}^4 \right) \quad \text{valid for small } \bar{r}, \quad (4.2)$$

where \bar{r} is given by (2.5).

These approximations assume that $\kappa \gg d$, which is typically not valid for high dimensions (see the discussion in Section 7 for an intuition). Also, the \bar{r} values corresponding to the text and gene expression data sets considered in this paper are in the mid-range rather than in the two extreme ranges of \bar{r} that are catered to by the above approximations. We obtain a more accurate approximation for κ as described below. With $A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}$, observe that $A_d(\kappa)$ is a ratio of

Bessel functions that differ in their order by just one. Fortunately there exists a continued fraction representation of $A_d(\kappa)$ (Watson, 1995) given by

$$A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)} = \frac{1}{\frac{d}{\kappa} + \frac{1}{\frac{d+2}{\kappa} + \dots}} \tag{4.3}$$

Letting $A_d(\kappa) = \bar{r}$ we can write (4.3) approximately as

$$\frac{1}{\bar{r}} \approx \frac{d}{\kappa} + \bar{r},$$

which gives the approximation,

$$\kappa \approx \frac{d\bar{r}}{1 - \bar{r}^2}.$$

We empirically found (see Section A.3 for details) that the quality of the above approximation can be improved by adding a correction term of $-\bar{r}^3/(1 - \bar{r}^2)$ to it. Thus we finally get

$$\hat{\kappa} = \frac{\bar{r}d - \bar{r}^3}{1 - \bar{r}^2}. \tag{4.4}$$

The approximation in (4.4) could perhaps be made even more accurate by adding other correction terms that are functions of \bar{r} and d .³ For other approximations of κ (including the derivations of (4.1) and (4.2)) and some related issues, the reader is referred to the detailed exposition in Dhillon and Sra (2003).

To properly assess the quality of our approximation and compare it with (4.1) and (4.2), first note that a particular value of \bar{r} may correspond to many different combinations of κ and d values. Thus, one needs to evaluate the accuracy of the approximations over the parts of the d - κ plane that are expected to be encountered in the target application domains. Section A.3 of the Appendix provides such an assessment by comparing performances over different slices of the d - κ plane and over a range of \bar{r} values. Below we simply compare the accuracies at a scattering of points on this plane via Table 1 which shows the actual numerical values of κ that the three approximations (4.1), (4.2), and (4.4) yielded at these points. The \bar{r} values shown in the table were computed using (2.5).

(d, \bar{r}, κ)	$\hat{\kappa} = \text{Eq. (4.1)}$	$\hat{\kappa} = \text{Eq. (4.2)}$	$\hat{\kappa} = \text{Eq. (4.4)}$
(10, 0.633668, 10)	12.2839	9.36921	10.1631
(100, 0.46945, 60)	93.2999	59.3643	60.0833
(500, 0.46859, 300)	469.506	296.832	300.084
(1000, 0.554386, 800)	1120.92	776.799	800.13

Table 1: Approximations $\hat{\kappa}$ for a sampling of κ and d values.

3. Note that if one wants a more accurate approximation, it is easier to use (4.4) as a starting point and then perform Newton-Raphson iterations for solving $A_d(\hat{\kappa}) - \bar{r} = 0$, since it is easy to evaluate $A'_d(\kappa) = 1 - A_d(\kappa)^2 - \frac{d-1}{\kappa} A_d(\kappa)$. However, for high-dimensional data, accurately computing $A_d(\kappa)$ can be quite slow compared to efficiently approximating $\hat{\kappa}$ using (4.4).

4.2 Analysis of Hard Assignments

In this subsection, we show that hard assignments should give a reasonable clustering in terms of the log-likelihood since they actually maximize a tight lower bound on the incomplete log-likelihood of the data. This result is applicable to any mixture model learning using EM, but the practical advantage in terms of lower computational demands seems to be more substantial when using mixtures of vMFs. The advantages are derived from the various facts outlined below:

- First, note that the partition function, $\sum_{l=1}^k \alpha_l f_l(\mathbf{x}_i|\theta_l)$, for every data point \mathbf{x}_i need not be computed for hard-assignments. This may not be a significant difference for several other models, but this is quite important for vMF distributions. Since the normalization terms $c_d(\kappa_h)$ in $f_h(\mathbf{x}_i|\theta_h)$ involve Bessel functions, any reasonable implementation of the algorithm has to employ high-precision representation to avoid under- and over-flows. As a result, computing the partition function is computationally intensive. For hard assignments, this computation is not required resulting in substantially faster running times. In particular, hard-assignments need $O(k)$ computations in high-precision per iteration simply to compute the normalization terms $c_d(\kappa_h), h = 1, \dots, k$. On the other hand, soft-assignments need $O(nk)$ computations in high-precision per iteration for all $f_l(\mathbf{x}_i|\theta_l)$ so that the partition function $\sum_{l=1}^k \alpha_l f_l(\mathbf{x}_i|\theta_l)$ and the probabilities $p(h|\mathbf{x}_i, \Theta)$ can be accurately computed.
- A second issue is regarding the space complexity. Since soft assignments compute all the conditional probabilities, the algorithm has to maintain nk floating point numbers at a desired level of precision. On the other hand, hard assignments only need to maintain the cluster assignments of each point, i.e., n integers. This issue can become critical for large data sets and large number of clusters.

Hence, a hard assignment scheme is often computationally more efficient and scalable both in terms of time and space complexity.

We begin by investigating certain properties of hard-assignments. Hard-assignments have been extensively used in the statistics (Coleman et al., 1999; McLachlan and Peel, 2000) as well as machine learning literature (Kearns et al., 1997; Banerjee et al., 2004). In statistics, the hard assignment approach is better known as classification maximum likelihood approach (McLachlan, 1982). Although soft-assignments are theoretically well motivated (Collins, 1997; Neal and Hinton, 1998), hard-assignments have not received much theoretical attention with some notable exceptions (Kearns et al., 1997). However, algorithms employing hard-assignments, being computationally more efficient especially for large data sets, are often typically more practical than algorithms that use soft-assignments. Hence it is worthwhile to examine the behavior of hard-assignments from a theoretical perspective. In the rest of this section, we formally study the connection between soft and hard-assignments in the EM framework.

The distribution q in (3.8) belongs to the class \mathcal{H} of probability distributions that assume probability value 1 for some mixture component and 0 for all others. In the hard assignment setting, the hidden random variables are restricted to have distributions that are members of \mathcal{H} . Since \mathcal{H} is a subset of all possible distributions on the events, for a typical mixture model the distribution following (3.7) will not belong to this subset. The important question is: Is there a way to optimally pick a distribution from \mathcal{H} , perform a regular M-step, and guarantee that the incomplete log-likelihood of the data does not decrease? Unfortunately, such a way may not exist in general. However, it is possible to reasonably lower bound the incomplete log-likelihood of the data using expectations

over an *optimal* distribution $q \in \mathcal{H}$, as elucidated below. Thus, clustering using hard-assignments essentially maximizes a lower bound on the incomplete log-likelihood.

We now show that the expectation over q is a reasonable lower bound on the incomplete log-likelihood of the data in the sense that the expectation over q is itself lower bounded by the expectation of the complete log-likelihood (3.2) over the distribution p given by (3.7). Further, we show that q as given by (3.8) gives the tightest lower bound among all distributions in \mathcal{H} .

Following the arguments of Neal and Hinton (1998), we introduce the function $F(\tilde{p}, \Theta)$ given by

$$F(\tilde{p}, \Theta) = E_{\tilde{p}}[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] + H(\tilde{p}), \quad (4.5)$$

where $H(\tilde{p})$ gives the Shannon entropy of a discrete distribution \tilde{p} . The E- and the M-steps of the EM algorithm can be shown to *alternately maximize* the function F . In the E-step, for a given value of Θ , the distribution \tilde{p} is chosen to maximize $F(\tilde{p}, \Theta)$ for that Θ , and, in the M-step, for a given value of \tilde{p} , the parameters Θ are estimated to maximize $F(\tilde{p}, \Theta)$ for the given \tilde{p} . Consider p given by (3.7). It can be shown (Neal and Hinton, 1998) that for a given Θ , this value of p is optimal, i.e. $p = \operatorname{argmax}_{\tilde{p}} F(\tilde{p}, \Theta)$. Then,

$$\begin{aligned} F(p, \Theta) &= E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] + H(p) \\ &= E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] - E_p[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)] \\ &= E_p \left[\ln \left(\frac{P(\mathcal{X}, \mathcal{Z}|\Theta)}{P(\mathcal{Z}|\mathcal{X}, \Theta)} \right) \right] = E_p[\ln P(\mathcal{X}|\Theta)] \\ &= \ln P(\mathcal{X}|\Theta). \end{aligned} \quad (4.6)$$

Since (3.7) gives the optimal choice of the distribution, the functional value of F is smaller for any other choice of \tilde{p} . In particular, if $\tilde{p} = q$ as in (3.8), we have

$$F(q, \Theta) \leq F(p, \Theta) = \ln P(\mathcal{X}|\Theta).$$

Since $H(q) = 0$, from (4.5) we have

$$E_q[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] \leq \ln P(\mathcal{X}|\Theta). \quad (4.7)$$

Thus, the expectation over q actually lower bounds the likelihood of the data. We go one step further to show that this is in fact a reasonably tight lower bound in the sense that the expectation over q is lower bounded by the expectation over p of the complete data log-likelihood. To this end, we first prove the following result.

Lemma 1 *If p is given by (3.7) and q is given by (3.8) then,*

$$E_p[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)] \leq E_q[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)].$$

Proof Let $h_i^* = \operatorname{argmax}_h p(h|\mathbf{x}_i, \Theta)$. Then, $p(h|\mathbf{x}_i, \Theta) \leq p(h_i^*|\mathbf{x}_i, \Theta), \forall h$. Using the definitions of p and q , we have

$$\begin{aligned}
 E_p[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)] &= \sum_{i=1}^n \sum_{h=1}^k p(h|\mathbf{x}_i, \Theta) \ln p(h|\mathbf{x}_i, \Theta) \\
 &\leq \sum_{i=1}^n \sum_{h=1}^k p(h|\mathbf{x}_i, \Theta) \ln p(h_i^*|\mathbf{x}_i, \Theta) \\
 &= \sum_{i=1}^n \ln p(h_i^*|\mathbf{x}_i, \Theta) \sum_{h=1}^k p(h|\mathbf{x}_i, \Theta) = \sum_{i=1}^n \ln p(h_i^*|\mathbf{x}_i, \Theta) \\
 &= \sum_{i=1}^n \sum_{h=1}^k q(h|\mathbf{x}_i, \Theta) \ln p(h|\mathbf{x}_i, \Theta) \\
 &= E_q[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)]. \quad \blacksquare
 \end{aligned}$$

Now, adding the incomplete data log-likelihood to both sides of the inequality proven above, we obtain

$$\begin{aligned}
 E_p[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)] + \ln P(\mathcal{X}|\Theta) &\leq E_q[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)] + \ln P(\mathcal{X}|\Theta), \\
 E_p[\ln(P(\mathcal{Z}|\mathcal{X}, \Theta)P(\mathcal{X}|\Theta))] &\leq E_q[\ln(P(\mathcal{Z}|\mathcal{X}, \Theta)P(\mathcal{X}|\Theta))], \\
 E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] &\leq E_q[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)]. \quad (4.8)
 \end{aligned}$$

From, (4.7) and (4.8), we infer

$$E_p[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] \leq E_q[\ln P(\mathcal{X}, \mathcal{Z}|\Theta)] \leq \ln P(\mathcal{X}|\Theta).$$

Let \tilde{q} be any other distribution in the class of distributions \mathcal{H} with $\tilde{q}(\tilde{h}_i|\mathbf{x}_i, \Theta) = 1$ and $\tilde{q}(h|\mathbf{x}_i, \Theta) = 0$ for $h \neq \tilde{h}_i$. Then,

$$\begin{aligned}
 E_{\tilde{q}}[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)] &= \sum_{i=1}^n \sum_{h=1}^k \tilde{q}(h|\mathbf{x}_i, \Theta) \ln p(h|\mathbf{x}_i, \Theta) = \sum_{i=1}^n \ln p(\tilde{h}_i|\mathbf{x}_i, \Theta) \\
 &\leq \sum_{i=1}^n \ln p(h_i^*|\mathbf{x}_i, \Theta) = \sum_{i=1}^n \sum_{h=1}^k q(h|\mathbf{x}_i, \Theta) \ln p(h|\mathbf{x}_i, \Theta) \\
 &= E_q[\ln P(\mathcal{Z}|\mathcal{X}, \Theta)].
 \end{aligned}$$

Hence, the choice of q as in (3.8) is optimal. This analysis forms the basis of the `hard-moVMF` algorithm presented in the next section.

5. Algorithms

The developments of the previous section naturally lead to two algorithms for clustering directional data. The algorithms are centered on soft and hard-assignment schemes and are titled `soft-moVMF` and `hard-moVMF` respectively. The `soft-moVMF` algorithm (Algorithm 1) estimates the parameters of the mixture model exactly following the derivations in Section 3 using EM. Hence, it assigns soft (or probabilistic) labels to each point that are given by the posterior probabilities of the components

Algorithm 1 soft-moVMF

Input: Set \mathcal{X} of data points on \mathbb{S}^{d-1}

Output: A soft clustering of \mathcal{X} over a mixture of k vMF distributions

Initialize all $\alpha_h, \mu_h, \kappa_h, h = 1, \dots, k$

repeat

{The E (Expectation) step of EM}

for $i = 1$ to n **do**

for $h = 1$ to k **do**

$$f_h(\mathbf{x}_i | \theta_h) \leftarrow c_d(\kappa_h) e^{\kappa_h \mu_h^T \mathbf{x}_i}$$

end for

for $h = 1$ to k **do**

$$p(h | \mathbf{x}_i, \Theta) \leftarrow \frac{\alpha_h f_h(\mathbf{x}_i | \theta_h)}{\sum_{l=1}^k \alpha_l f_l(\mathbf{x}_i | \theta_l)}$$

end for

end for

{The M (Maximization) step of EM}

for $h = 1$ to k **do**

$$\alpha_h \leftarrow \frac{1}{n} \sum_{i=1}^n p(h | \mathbf{x}_i, \Theta)$$

$$\mu_h \leftarrow \sum_{i=1}^n \mathbf{x}_i p(h | \mathbf{x}_i, \Theta)$$

$$\bar{r} \leftarrow \|\mu_h\| / (n\alpha_h)$$

$$\mu_h \leftarrow \mu_h / \|\mu_h\|$$

$$\kappa_h \leftarrow \frac{\bar{r}^d - \bar{r}^3}{1 - \bar{r}^2}$$

end for

until convergence

of the mixture conditioned on the point. On termination, the algorithm gives the parameters $\Theta = \{\alpha_h, \mu_h, \kappa_h\}_{h=1}^k$ of the k vMF distributions that model the data set \mathcal{X} , as well as the *soft-clustering*, i.e., the posterior probabilities $p(h | \mathbf{x}_i, \Theta)$, for all h and i .

The hard-moVMF algorithm (Algorithm 2) estimates the parameters of the mixture model using a hard assignment, or, *winner takes all* strategy. In other words, we do the assignment of the points based on a derived posterior distribution given by (3.8). After the hard assignments in every iteration, each point *belongs* to a single cluster. As before, the updates of the component parameters are done using the posteriors of the components, given the points. The crucial difference in this case is that the posterior probabilities are allowed to take only binary (0/1) values. Upon termination, Algorithm 2 yields a hard clustering of the data and the parameters $\Theta = \{\alpha_h, \mu_h, \kappa_h\}_{h=1}^k$ of the k vMFs that model the input data set \mathcal{X} .

5.1 Revisiting Spherical Kmeans

In this section we show that upon enforcing certain restrictive assumptions on the generative model, the `spkmeans` algorithm (Algorithm 3) can be viewed as a special case of both the `soft-moVMF` and `hard-moVMF` algorithms.

More precisely, assume that in our mixture of vMFs, the priors of all the components are equal, i.e., $\alpha_h = 1/k$ for all h . Further assume that all the components have (equal) infinite concentration parameters, i.e., $\kappa_h = \kappa \rightarrow \infty$ for all h . Under these assumptions the E-step in the `soft-moVMF`

Algorithm 2 hard-moVMF

Input: Set \mathcal{X} of data points on \mathbb{S}^{d-1}
Output: A disjoint k -partitioning of \mathcal{X}

 Initialize all $\alpha_h, \mu_h, \kappa_h, h = 1, \dots, k$
repeat

{The Hardened E (Expectation) step of EM}

for $i = 1$ to n **do**
for $h = 1$ to k **do**

$$f_h(\mathbf{x}_i | \theta_h) \leftarrow c_d(\kappa_h) e^{\kappa_h \mu_h^T \mathbf{x}_i}$$

end for
for $h = 1$ to k **do**

$$q(h | \mathbf{x}_i, \Theta) \leftarrow \begin{cases} 1, & \text{if } h = \arg \max_{h'} \alpha_{h'} f_{h'}(\mathbf{x}_i | \theta_{h'}) \\ 0, & \text{otherwise.} \end{cases}$$

end for
end for

{The M (Maximization) step of EM}

for $h = 1$ to k **do**

$$\alpha_h \leftarrow \frac{1}{n} \sum_{i=1}^n q(h | \mathbf{x}_i, \Theta)$$

$$\mu_h \leftarrow \sum_{i=1}^n \mathbf{x}_i q(h | \mathbf{x}_i, \Theta)$$

$$\bar{r} \leftarrow \|\mu_h\| / (n\alpha_h)$$

$$\mu_h \leftarrow \mu_h / \|\mu_h\|$$

$$\kappa_h \leftarrow \frac{\bar{r}d - \bar{r}^3}{1 - \bar{r}^2}$$

end for
until convergence.

algorithm reduces to assigning a point to its *nearest* cluster, where nearness is computed as a cosine similarity between the point and the cluster representative. Thus, a point \mathbf{x}_i will be assigned to cluster $h^* = \arg \max_h \mathbf{x}_i^T \mu_h$, since

$$p(h^* | \mathbf{x}_i, \Theta) = \lim_{\kappa \rightarrow \infty} \frac{e^{\kappa \mathbf{x}_i^T \mu_{h^*}}}{\sum_{h=1}^k e^{\kappa \mathbf{x}_i^T \mu_h}} = 1,$$

and $p(h | \mathbf{x}_i, \Theta) \rightarrow 0$, as $\kappa \rightarrow \infty$ for all $h \neq h^*$.

To show that spkmeans can also be seen as a special case of the hard-moVMF, in addition to assuming the priors of the components to be equal, we further assume that the concentration parameters of all the components are equal, i.e., $\kappa_h = \kappa$ for all h . With these assumptions on the model, the estimation of the common concentration parameter becomes unessential since the hard assignment will depend only on the value of the cosine similarity $\mathbf{x}_i^T \mu_h$, and hard-moVMF reduces to spkmeans.

In addition to the abovementioned algorithms, we report experimental results on another algorithm fskmeans (Banerjee and Ghosh, 2002) that belongs to the same class in the sense that, like spkmeans, it can be derived from the mixture of vMF models with some restrictive assumptions. In fskmeans, the centroids of the mixture components are estimated as in hard-moVMF. The κ value

Algorithm 3 spkmeans

Input: Set \mathcal{X} of data points on \mathbb{S}^{d-1} **Output:** A disjoint k -partitioning $\{\mathcal{X}_h\}_{h=1}^k$ of \mathcal{X} Initialize $\mu_h, h = 1, \dots, k$ **repeat**

{The E (Expectation) step of EM}

 Set $\mathcal{X}_h \leftarrow \emptyset, h = 1, \dots, k$ **for** $i = 1$ to n **do** $\mathcal{X}_h \leftarrow \mathcal{X}_h \cup \{\mathbf{x}_i\}$ where $h = \operatorname{argmax}_{h'} \mathbf{x}_i^T \mu_{h'}$ **end for**

{The M (Maximization) step of EM}

for $h = 1$ to k **do**
$$\mu_h \leftarrow \frac{\sum_{\mathbf{x} \in \mathcal{X}_h} \mathbf{x}}{\|\sum_{\mathbf{x} \in \mathcal{X}_h} \mathbf{x}\|}$$
 end for**until** *convergence*.

for a component is *explicitly set* to be inversely proportional to the number of points in the cluster corresponding to that component. This explicit choice simulates a frequency sensitive competitive learning that implicitly prevents the formation of null clusters, a well-known problem in regular kmeans (Bradley et al., 2000).

6. Experimental Results

We now offer some experimental validation to assess the quality of clustering results achieved by our algorithms. We compare the following four algorithms on numerous data sets.

1. Spherical KMeans (Dhillon and Modha, 2001)—spkmeans.
2. Frequency Sensitive Spherical KMeans (Banerjee and Ghosh, 2002)—fskmeans.
3. moVMF based clustering using hard assignments (Section 3)—hard-moVMF.
4. moVMF based clustering using soft assignments (Section 3)—soft-moVMF.

It has already been established that kmeans using Euclidean distance performs much worse than spkmeans for text data (Strehl et al., 2000), so we do not consider it here. Generative model based algorithms that use mixtures of Bernoulli or multinomial distributions, which have been shown to perform well for text data sets, have also not been included in the experiments. This exclusion is done as a recent empirical study over 15 text data sets showed that simple versions of vMF mixture models (with κ constant for all clusters) outperform the multinomial model except for only one data set (Classic3), and the Bernoulli model was inferior for all data sets (Zhong and Ghosh, 2003b).

6.1 Data Sets

The data sets that we used for empirical validation and comparison of our algorithms were carefully selected to represent some typical clustering problems. We also created various subsets of some

of the data sets for gaining greater insight into the nature of clusters discovered or to model some particular clustering scenario (e.g., balanced clusters, skewed clusters, overlapping clusters etc.). We drew our data from five sources: Simulation, Classic3, Yahoo News, CMU 20 Newsgroup and Yeast Gene Expressions. For all the text document data sets, the toolkit MC (Dhillon et al., 2001) was used for creating a high-dimensional vector space model that each of the four algorithms utilized. MATLAB code was used to render the input as a vector space for both the simulated and gene-expression data sets.

- **Simulation.** We use simulated data to verify that the discrepancy between computed values of the parameters and their true values is small. Our simulated data serves the principal purpose of validating the “correctness” of our implementations. We used a slight modification of the algorithm given by Wood (1994) to generate a set of data points following a given vMF distribution. We describe herein, two synthetic data sets. The first data set **small-mix** is 2-dimensional and is used to illustrate soft-clustering. The second data set **big-mix** is a high-dimensional data set that could serve as a model for real world text data sets. Let the triple (n, d, k) denote the number of sample points, the dimensionality of a sample point and the number of clusters respectively.
 1. **small-mix:** This data has $(n, d, k) = (50, 2, 2)$. The mean direction of each component is a random unit vector. Each component has $\kappa = 4$.
 2. **big-mix:** This data has $(n, d, k) = (5000, 1000, 4)$. The mean direction of each component is a random unit vector, and the κ values of the components are 650.98, 266.83, 267.83, and 612.88. The mixing weights for each component are 0.251, 0.238, 0.252, and 0.259.
- **Classic3.** Classic3 is a well known collection of documents. It is an easy data set to cluster since it contains documents from three well-separated sources. Moreover, the intrinsic clusters are largely balanced.
 1. **Classic3:** This corpus contains 3893 documents, among which 1400 CRANFIELD documents are from aeronautical system papers, 1033 MEDLINE documents are from medical journals, and 1460 CISI documents are from information retrieval papers. The particular vector space model used had a total of 4666 features (words). Thus each document, after normalization, is represented as a unit vector in a 4666-dimensional space.
 2. **Classic300:** Classic300 is a subset of the Classic3 collection and has 300 documents. From each category of Classic3, we picked 100 documents at random to form this particular data set. The dimensionality of the data was 5471.⁴
 3. **Classic400:** Classic400 is a subset of Classic3 that has 400 documents. This data set has 100 randomly chosen documents from the MEDLINE and CISI categories and 200 randomly chosen documents from the CRANFIELD category. This data set is specifically designed to create unbalanced clusters in an otherwise easily separable and balanced data set. The dimensionality of the data was 6205.

4. Note that the dimensionality in Classic300 is larger than the that of Classic3. Although the same options were used in the MC toolkit for word pruning, due to very different words distributions, fewer words got pruned for Classic300 in the ‘too common’ or ‘too rare’ categories.

- **Yahoo News (K-series).** This compilation has 2340 Yahoo news articles from 20 different categories. The underlying clusters in this data set are highly skewed in terms of the number of documents per cluster, with sizes ranging from 9 to 494. The skewness presents additional challenges to clustering algorithms.
- **CMU Newsgroup.** The CMU Newsgroup data set is a well known compilation of documents (Newsgroups). We tested our algorithms on not only the original data set, but on a variety of subsets with differing characteristics to explore and understand the behavior of our algorithms.
 1. **News20:** This standard data set is a collection of 19,997 messages, gathered from 20 different USENET newsgroups. One thousand messages are drawn from the first 19 newsgroups, and 997 from the twentieth. The headers for each of the messages are then removed to avoid biasing the results. The particular vector space model used had 25924 words. News20 embodies the features characteristic of a typical text data set—high-dimensionality, sparsity and significantly overlapping clusters.
 2. **Small-news20:** We formed this set by selecting 2000 messages from original News20 data set. We randomly selected 100 messages from each category in the original data set. Hence this data set has balanced classes (though there may be overlap). The dimensionality of the data was 13406.
 3. **Same-100/1000** is a collection of 100/1000 messages from 3 very similar newsgroups: comp.graphics, comp.os.ms-windows, comp.windows.x.
 4. **Similar-100/1000** is a collection of 100/1000 messages from 3 somewhat similar newsgroups: talk.politics.guns, talk.politics.mideast, talk.politics.misc.
 5. **Different-100/1000** is a collection of 100/1000 messages from 3 very different newsgroups: alt.atheism, rec.sport.baseball, sci.space.
- **Yeast Gene Expressions.** Gene-expression data was selected to offer a clustering domain different from text analysis. As previously motivated, the use of Pearson correlation for the analysis of gene expression data is common, so a directional model is well-suited. Coincident to this domain are the difficulties of cluster validation because of the unavailability of true labels. Such difficulties make the gene expression data a more challenging and perhaps a more rewarding domain for clustering.

Gene expression data is presented as a matrix of genes (rows) by expression values (columns). The expression vectors are constructed using DNA microarray experiments. We used a subset of the Rosetta Inpharmatics yeast gene expression set (Hughes et al., 2000). The original data set consists of 300 experiments measuring expression of 6,048 yeast genes. Out of these we selected a subset of 996 genes for clustering (Dhillon et al., 2002b). For each of the 996 genes the 300-element expression vector was normalized to have unit Euclidean (L_2) norm.

6.2 Methodology

Except for the gene expression data set, performance of the algorithms on all the data sets has been analyzed using *mutual information* (MI) between the cluster and class labels. For gene data, due to the absence of true labels, we have to take recourse to reporting some internal figures of merit. We defer a discussion of the same to Section 6.7.

The MI gives the amount of statistical similarity between the cluster and class labels (Cover and Thomas, 1991). If X is a random variable for the cluster assignments and Y is a random variable for the pre-existing labels on the same data, then their MI is given by $I(X; Y) = E[\ln \frac{p(X,Y)}{p(X)p(Y)}]$ where the expectation is computed over the joint distribution of (X, Y) estimated from a particular clustering of the data set under consideration. For `soft-moVMF` we “harden” the clustering produced by labeling a point with the cluster label for which it has the highest value of posterior probability (ties broken arbitrarily), in order to evaluate MI. Note that variants of MI have been used to evaluate clustering algorithms by several researchers. Meilă (2003) used a related concept called variation of information to compare clusterings. An MDL-based formulation that uses the MI between cluster assignments and class labels was proposed by Dom (2001).

All results reported herein have been averaged over 10 runs. All algorithms were started with the same random initialization to ensure fairness of comparison. Each run was started with a *different* random initialization. However, no algorithm was restarted within a given run and all of them were allowed to run to completion. Since the standard deviations of MI were reasonably small for all algorithms, to reduce clutter, we have chosen to omit a display of error bars in our plots. Also, for practical reasons, the estimate of κ was upper bounded by a large number (10^4 , in this case) in order to prevent numeric overflows. For example, during the iterations, if a cluster has only one point, the estimate of κ will be infinity (a divide by zero error). Upper bounding the estimate is similar in flavor to ensuring the estimated covariance of a multi-variate Gaussian in a mixture of Gaussians to be non-singular.

6.3 Simulated Data Sets

First, to build some intuition and confidence in the working of our vMF based algorithms we exhibit relevant details of `soft-moVMF`'s behavior on the small-mix data set shown in Figure 1(a).

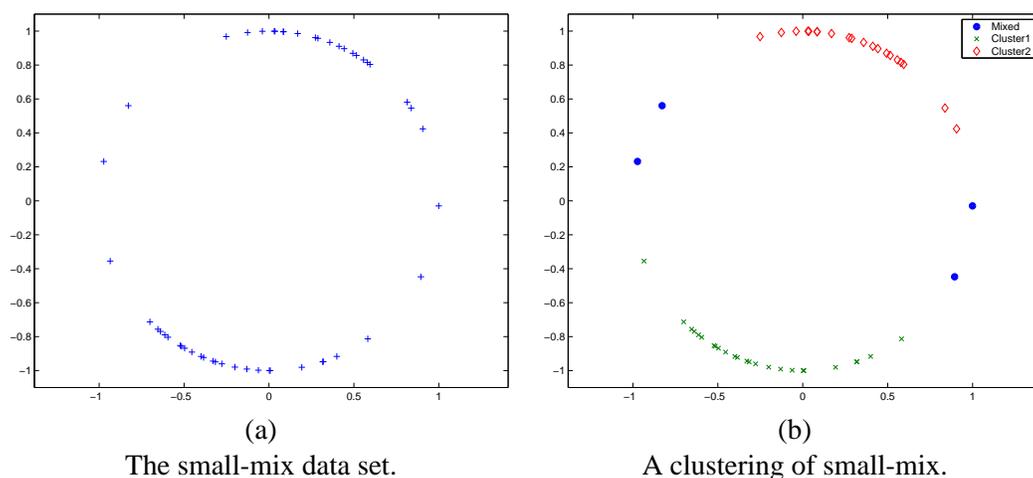


Figure 1: Small-mix data set and its clustering by `soft-moVMF`.

The clustering produced by our soft cluster assignment algorithm is shown in Figure 1(b). The four points (taken clockwise) marked with solid circles have cluster labels $(0.15, 0.85)$, $(0.77, 0.23)$, $(.82, .18)$ and $(.11, .89)$, where a cluster label $(p, 1 - p)$ for a point means that the point has proba-

bility p of belonging to Cluster 1 and probability $1 - p$ of belonging to Cluster 2. All other points are categorized to belong to a single cluster by ignoring small (less than 0.10) probability values.

The confusion matrix, obtained by “hardening” the clustering produced by `soft-moVMF` for the small-mix data set is $\begin{bmatrix} 26 & 1 \\ 0 & 23 \end{bmatrix}$. As is evident from this confusion matrix, the clustering performed by `soft-moVMF` is excellent, though not surprising, since small-mix is a data set with well-separated clusters. Further testimony to `soft-moVMF`’s performance is served by Table 2, which shows the discrepancy between true and estimated parameters for the small-mix collection.

Cluster	μ	$\hat{\mu}$	κ	$\hat{\kappa}$	α	$\hat{\alpha}$
1	(-0.251, -0.968)	(-0.279, -0.960)	4	3.78	0.48	0.46
2	(0.399, 0.917)	(0.370, 0.929)	4	3.53	0.52	0.54

Table 2: True and estimated parameters for small-mix using `soft-moVMF`.

In the table μ, κ, α represent the true parameters and $\hat{\mu}, \hat{\kappa}, \hat{\alpha}$ represent the estimated parameters. We can see that even in the presence of a limited number of data points in the small-mix data set (50 points), the estimated parameters approximate the true parameters quite well.

Before moving onto real data sets let us briefly look at the behavior of the algorithms on the larger data set big-mix. On calculating MI as described previously we found that all the algorithms performed similarly with MI values close to one. We attribute this good performance of all the

$\min \mu^T \hat{\mu}$	$\text{avg } \mu^T \hat{\mu}$	$\max \frac{ \kappa - \hat{\kappa} }{ \kappa }$	$\text{avg } \frac{ \kappa - \hat{\kappa} }{ \kappa }$	$\max \frac{ \alpha - \hat{\alpha} }{ \alpha }$	$\text{avg } \frac{ \alpha - \hat{\alpha} }{ \alpha }$
0.994	0.998	0.006	0.004	0.002	0.001

Table 3: Performance of `soft-moVMF` on big-mix data set.

algorithms to the availability of a sufficient number of data points and similar sized clusters. For reference Table 3 offers numerical evidence about the performance of `soft-moVMF` on the big-mix data set.

6.4 Classic3 Family of Data Sets

Table 4 shows typical confusion matrices obtained for the full Classic3 data set. We observe that the performance of all the algorithms is quite similar and there is no added advantage yielded by using the general `moVMF` model as compared to the other algorithms. This observation can be explained by noting that the clusters of Classic3 are well separated and have a sufficient number of documents. For this clustering `hard-moVMF` yielded κ values of (732.13, 809.53, 1000.04), while `soft-moVMF` reported κ values of (731.55, 808.21, 1002.95).

Table 5 shows the confusion matrices obtained for the Classic300 data set. Even though Classic300 is well separated, the small number of documents per cluster makes the problem somewhat difficult for `fskmeans` and `spkmeans`, while `hard-moVMF` has a much better performance due to its model flexibility. The `soft-moVMF` algorithm performs appreciably better than the other three algorithms.

It seems that the low number of documents does not pose a problem for `soft-moVMF` and it ends up getting an almost perfect clustering for this data set. Thus in this case, despite the low number

fskmeans			spkmeans			hard-moVMF			soft-moVMF		
med	cisi	cran									
1019	0	0	1019	0	0	1018	0	0	1019	0	1
1	6	1386	1	6	1386	2	6	1387	1	4	1384
13	1454	12	13	1454	12	13	1454	11	13	1456	13

Table 4: Comparative confusion matrices for 3 clusters of Classic3 (rows represent clusters).

fskmeans			spkmeans			hard-moVMF			soft-moVMF		
med	cisi	cran	med	cisi	cran	med	cisi	cran	med	cisi	cran
29	38	22	29	38	22	3	72	1	0	98	0
31	27	38	31	27	38	62	28	17	99	2	0
40	35	40	40	35	40	35	0	82	1	0	100

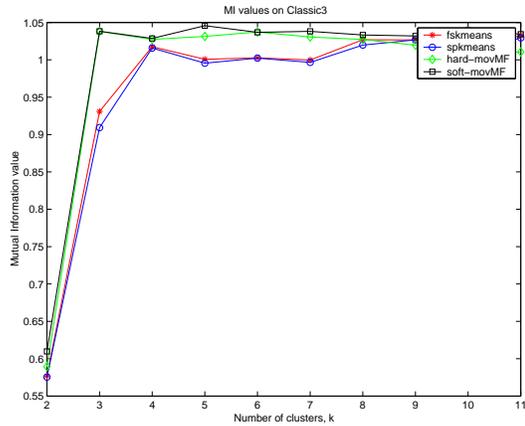
Table 5: Comparative confusion matrices for 3 clusters of Classic300.

of points per cluster, the superior modeling power of our moVMF based algorithms prevents them from getting trapped in inferior local-minima as compared to the other algorithms—resulting in a better clustering.

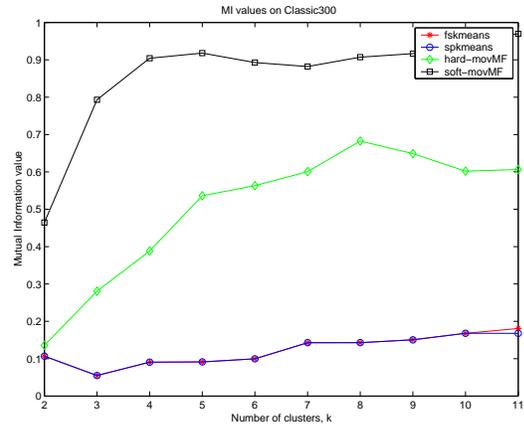
The confusion matrices obtained for the Classic400 data set are displayed in Table 6. The behavior of the algorithms for this data set is quite interesting. As before, due to the small number of documents per cluster, fskmeans and spkmeans give a rather mixed confusion matrix. The hard-moVMF algorithm gets a significant part of the bigger cluster correctly and achieves some amount of separation between the two smaller clusters. The soft-moVMF algorithm exhibits a somewhat intriguing behavior. It splits the bigger cluster into two, relatively pure segments, and merges the smaller two into one cluster. When 4 clusters are requested from soft-moVMF, it returns 4 very pure clusters (not shown in the confusion matrices) two of which are almost equal sized segments of the bigger cluster.

An engaging insight into the working of the algorithms is provided by considering their clustering performance when they are requested to produce greater than the “natural” number of clusters. In Table 7 we show the confusion matrices resulting from 5 clusters of the Classic3 corpus. The matrices suggest that the moVMF algorithms have a tendency of trying to maintain larger clusters intact as long as possible, and breaking them into reasonably pure and comparably sized parts when they absolutely must. This behavior of our moVMF algorithms coupled with the observations in Table 6, suggest a clustering method in which one could generate a slightly higher number of clusters than required, and then agglomerate them appropriately.

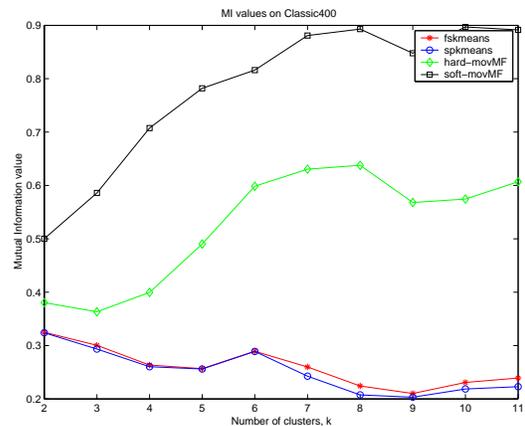
The MI plots for the various Classic3 data sets are given in Figures 2(a)-(c). For the full Classic3 data set (Figure 2(a)), all the algorithms perform almost similarly at the true number of clusters. However, as the number of clusters increases, soft-moVMF seems to outperform the others by a significant margin. For Classic300 (Figure 2(b)) and Classic400 (Figure 2(c)), soft-moVMF seems to significantly outperform the other algorithms. In fact, for these two data sets, soft-moVMF performs substantially better than the other three, even at the correct number of clusters. Among the other three, hard-moVMF seems to perform better than spkmeans and fskmeans across the range of clusters.



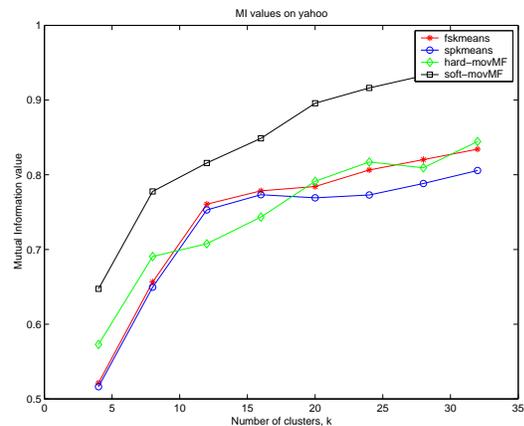
(a) Comparison of MI values for Classic3.



(b) Comparison of MI values for Classic300.



(c) Comparison of MI values for Classic400.



(d) Comparison of MI values for Yahoo20.

Figure 2: Comparison of the algorithms for the Classic3 data sets and the Yahoo News data set.

fskmeans			spkmeans			hard-moVMF			soft-moVMF		
med	cisi	cran	med	cisi	cran	med	cisi	cran	med	cisi	cran
27	16	55	27	17	54	56	28	20	0	0	91
51	83	12	51	82	12	44	72	14	82	99	2
23	1	132	23	1	133	1	0	165	19	1	106

Table 6: Comparative confusion matrices for 3 clusters of Classic400.

fskmeans			spkmeans			hard-moVMF			soft-moVMF		
med	cisi	cran	med	cisi	cran	med	cisi	cran	med	cisi	cran
2	4	312	2	4	323	3	5	292	0	1	1107
8	520	10	8	512	9	511	1	0	5	1455	14
5	936	6	5	944	6	514	1	0	526	2	1
1018	0	1	1018	0	1	0	2	1093	501	0	0
0	0	1069	0	0	1059	5	1451	13	1	2	276

Table 7: Comparative confusion matrices for 5 clusters of Classic3.

6.5 Yahoo News Data Set

The Yahoo News data set is a relatively difficult data set for clustering since it has a fair amount of overlap among its clusters and the number of points per cluster is low. In addition, the clusters are highly skewed in terms of their comparative sizes.

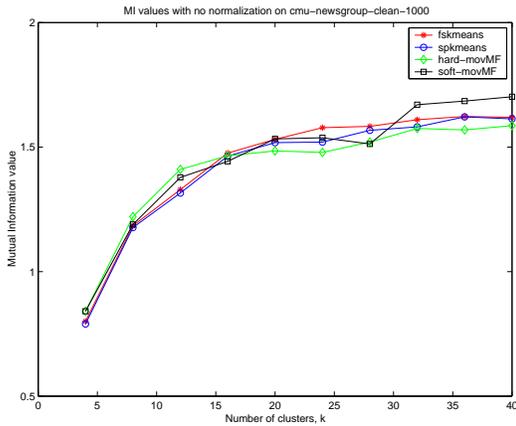
Results for the different algorithms can be seen in Figure 2(d). Over the entire range, *soft-moVMF* consistently performs better than the other algorithms. Even at the correct number of clusters $k = 20$, it performs significantly better than the other algorithms.

6.6 CMU Newsgroup Family of Data Sets

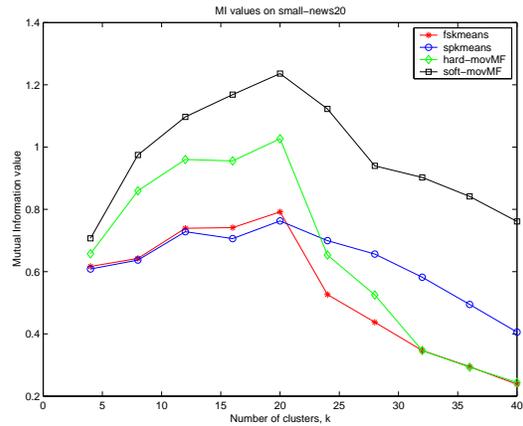
Now we discuss clustering performance of the four algorithms on the CMU Newsgroup data sets. Figure 3(a) shows the MI plots for the full News20 data set. All the algorithms perform similarly until the true number of clusters after which *soft-moVMF* and *spkmeans* perform better than the others. We do not notice any interesting differences between the four algorithms from this Figure.

Figure 3(b) shows MI plots for the Small-News20 data set and the results are of course different. Since the number of documents per cluster is small (100), as before *spkmeans* and *fskmeans* do not perform that well, even at the true number of clusters, whereas *soft-moVMF* performs considerably better than the others over the entire range. Again, *hard-moVMF* exhibits good MI values until the true number of clusters, after which it falls sharply. On the other hand, for the data sets that have a reasonably large number of documents per cluster, another kind of behavior is usually observed. All the algorithms perform quite similarly until the true number of clusters, after which *soft-moVMF* performs significantly better than the other three. This behavior can be observed in Figures 3(d), 3(f) and 4(b). We note that the other three algorithms perform quite similarly over the entire range of clusters. We also observe that for an easy data set like Different-1000, the MI values peak at the true number of clusters, whereas for a more difficult data set such as Similar-1000 the MI values increase as the clusters get further refined. This behavior is expected since the clusters in Similar-1000 have much greater overlap than those in Different-1000.

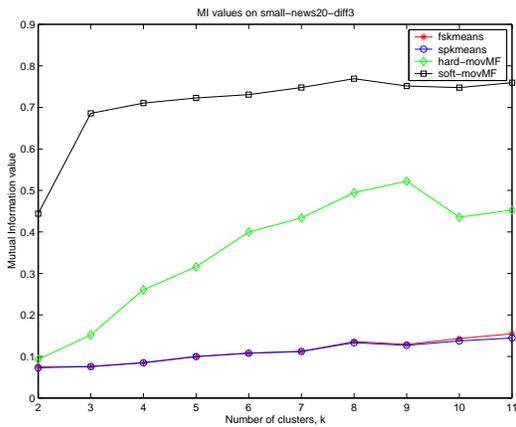
CLUSTERING WITH VON MISES-FISHER DISTRIBUTIONS



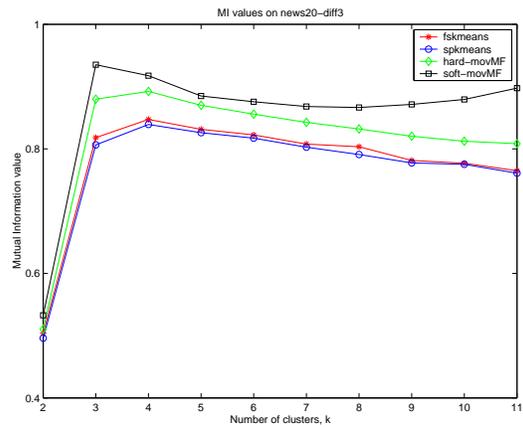
(a) Comparison of MI values for News20.



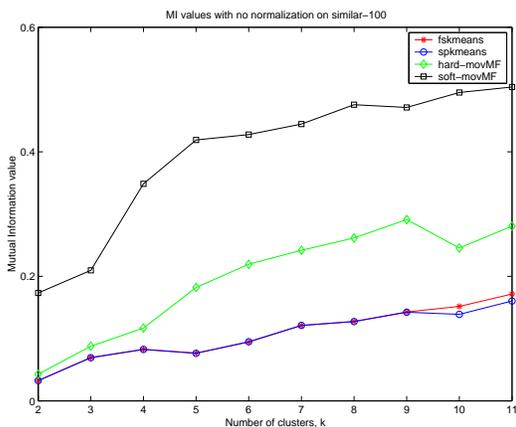
(b) Comparison of MI values for Small-news20.



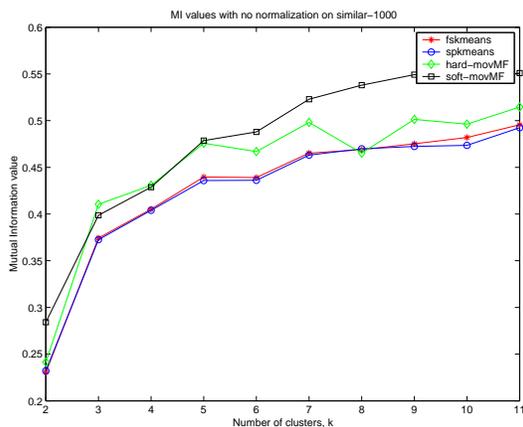
(c) Comparison of MI values for Different-100.



(d) Comparison of MI values for Different-1000.

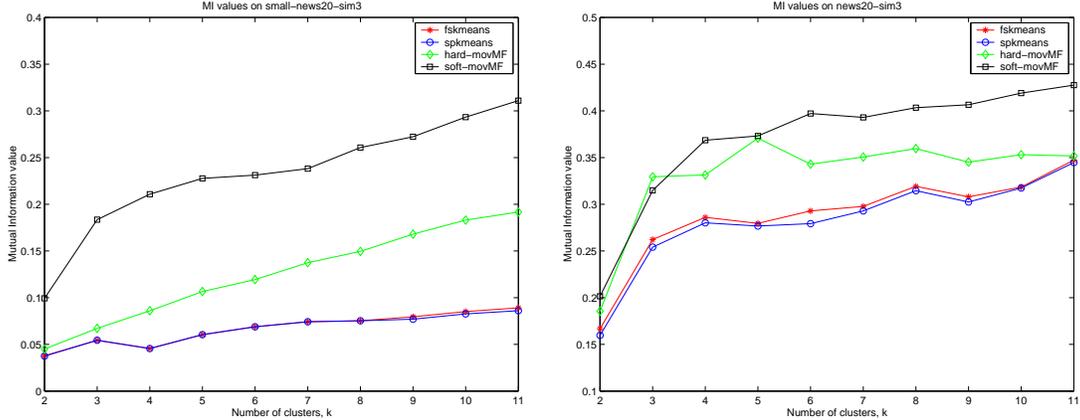


(e) Comparison of MI values for Similar-100.



(f) Comparison of MI values for Similar-1000.

Figure 3: Comparison of the algorithms for the CMU Newsgroup and some subsets.



(a) Comparison of MI values for Same-100. (b) Comparison of MI values for Same-1000.

Figure 4: Comparison of the algorithms for more subsets of CMU Newsgroup data.

6.7 Yeast Gene Expression Data Set

The gene data set that we consider differs from text data in two major aspects. First, the data can have negative values, and second, we do not know the true labels for the data points.

Owing to the absence of true cluster labels for the data points, we evaluate the clusterings by computing certain internal figures of merit. These internal measures have been earlier employed for evaluating clustering of genes (e.g., Sharan and Shamir, 2000). Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be the set of data that is clustered into disjoint clusters $\mathcal{X}_1, \dots, \mathcal{X}_k$. Let μ_j denote the mean vector of the j -th cluster ($1 \leq j \leq k$). The homogeneity of the clustering is measured by

$$H_{avg} = \frac{1}{|\mathcal{X}|} \sum_{j=1}^k \sum_{x \in \mathcal{X}_j} \frac{x^T \mu_j}{\|x\| \|\mu_j\|}. \quad (6.1)$$

As can easily be seen, a higher homogeneity means that the individual elements of each cluster are quite similar to the cluster representative. We also take note of the minimum similarity

$$H_{min} = \min_{\substack{1 \leq j \leq k \\ x \in \mathcal{X}_j}} \frac{x^T \mu_j}{\|x\| \|\mu_j\|}. \quad (6.2)$$

Both H_{avg} and H_{min} provide a measure of the intra-cluster similarity. We now define the inter-cluster separation as

$$S_{avg} = \frac{1}{\sum_{i \neq j} |\mathcal{X}_i| |\mathcal{X}_j|} \sum_{i \neq j} |\mathcal{X}_i| |\mathcal{X}_j| \frac{\mu_i^T \mu_j}{\|\mu_i\| \|\mu_j\|}. \quad (6.3)$$

We also take note of the maximum inter-cluster similarity

$$S_{max} = \max_{i \neq j} \frac{\mu_i^T \mu_j}{\|\mu_i\| \|\mu_j\|}. \quad (6.4)$$

It is easily seen that for a “good” clustering S_{avg} and S_{max} should be low.

Recently, researchers (Segal et al., 2003; Lee et al., 2004) have started looking at supervised methods of evaluating the gene clustering results using public genome databases such as the Kyoto Encyclopedia of Genes and Genomes (KEGG) and the gene ontology (GO). As of now, the evaluation techniques are still evolving and there is no consensus on how to best use the databases. For example, it is becoming clear that a pairwise precision-recall analysis of gene pairs may not be useful since the databases are currently incomplete due to lack of knowledge about all genes. In the recent past, progress has been made in terms of supervised evaluation and online tools such as GoMiner (GoMiner03) have been developed. As future work, we would like to evaluate the performance of our proposed algorithms using such tools.

Figure 5 shows the various cluster quality figures of merit as computed for clusters of our gene expression data. A fact that one immediately observes is that `hard-moVMF` consistently performs better than all the other algorithms. This comes as somewhat of a surprise, because in almost all other data sets, `soft-moVMF` performs better (though, of course, the measures of evaluation are different for gene data as compared to the other data sets that we considered). Note that the figures of merit for `soft-moVMF` are computed after “hardening” the clustering results that it produced.

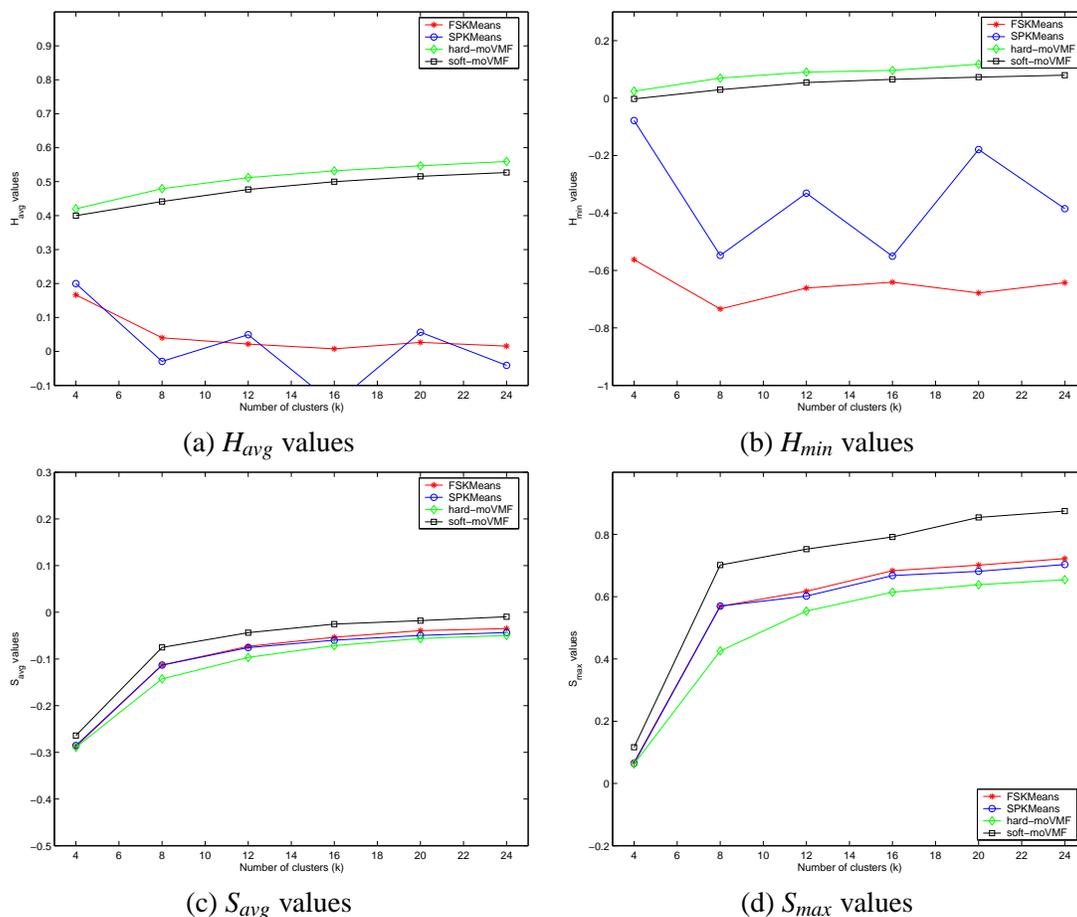


Figure 5: Measures of cluster quality for gene data.

We see from Figure 5(a) that both `hard-moVMF` and `soft-moVMF` yield clusters that are much more homogeneous than those furnished by `fskmeans` and `spkmeans`. The inter-cluster similarities, as measured by S_{avg} and S_{max} are again the lowest for `hard-moVMF`, thereby indicating that `hard-moVMF` gives the best separated clusters of all the four algorithms. Though the inter-cluster similarities do not differ that much between the four algorithms, `soft-moVMF` seems to be forming clusters with higher inter-cluster similarity than other algorithms. We could explain this behavior of `soft-moVMF` by noting that it tends to form overlapping clusters (because of soft-assignments) and those clusters remain closer even after hardening. Since H_{avg} essentially measures the average cosine similarity, we note that using our `moVMF` based algorithms, we are able to achieve clusters that are more coherent and better separated—a fact that could be attributed to the richer model employed by our algorithms. An inescapable observation is that our `vMF` based algorithms obtain a better average cosine similarity than `spkmeans`, implying that the richer `vMF` model allows them to escape the local minima that trap `spkmeans`.

6.8 Running Time

This section shows a brief report of the running time differences between `hard-moVMF` and `soft-moVMF`. Table 8 shows these comparisons. These running time experiments were performed on an AMD Athlon based computer running the Linux operating system. From Table 8 we see that `hard-moVMF`

Clusters	Classic300	Classic3	News20
3	0.39s/11.56s	3.03s/109.87s	10.18s/619.68s
5	0.54s/17.99s	3.59s/163.09s	14.05s/874.13s
10	-	-	18.9s/1512s
20	-	-	29.08s/3368s

Table 8: Running time comparison between `hard-moVMF` and `soft-moVMF`. The times are indicated in the format “`hard-moVMF/soft-moVMF`”.

runs much faster than `soft-moVMF`, and this difference becomes even greater when the number of clusters desired becomes higher.

7. Discussion

The mixture of `vMF` distributions gives a parametric model-based generalization of the widely used cosine similarity measure. As discussed in Section 5.1, the spherical `kmeans` algorithm that uses cosine similarity arises as a special case of EM on mixture of `vMFs` when, among other things, the concentration κ of all the distributions is held constant. Interestingly, an alternative and more formal connection can be made from an information geometry viewpoint (Amari, 1995). More precisely, consider a data set that has been sampled following a `vMF` distribution with a given κ , say $\kappa = 1$. Assuming the Fisher-Information matrix is identity, the Fisher kernel similarity (Jaakkola and Haussler, 1999) corresponding to the `vMF` distribution is given by

$$\begin{aligned}
 K(\mathbf{x}_i, \mathbf{x}_j) &= (\nabla_{\boldsymbol{\mu}} \ln f(\mathbf{x}_i | \boldsymbol{\mu}))^T (\nabla_{\boldsymbol{\mu}} \ln f(\mathbf{x}_j | \boldsymbol{\mu})) \quad (\text{see (2.1)}) \\
 &= (\nabla_{\boldsymbol{\mu}} (\boldsymbol{\mu}^T \mathbf{x}_i))^T (\nabla_{\boldsymbol{\mu}} (\boldsymbol{\mu}^T \mathbf{x}_j)) = \mathbf{x}_i^T \mathbf{x}_j,
 \end{aligned}$$

which is exactly the cosine similarity. This provides a theoretical justification for a long-practiced approach in the information retrieval community.

In terms of performance, the magnitude of improvement shown by the `soft-movMF` algorithm for the difficult clustering tasks was surprising, especially since for low-dimensional non-directional data, the improvements using a soft, EM-based `kmeans` or fuzzy `kmeans` over the standard hard-assignment based versions are often quite minimal. In particular, we were curious regarding a couple of issues: (i) why is `soft-movMF` performing substantially better than `hard-movMF`, even though the final probability values obtained by `soft-movMF` are actually very close to 0 and 1; and (ii) why is `soft-movMF`, which needs to estimate more parameters, doing better even when there are insufficient number of points relative to the dimensionality of the space.

It turns out that both these issues can be understood by taking a closer look at how `soft-movMF` converges. In all our experiments, we initialized κ to 10, and the initial centroids to small random perturbations of the global centroid. Hence, for `soft-movMF`, the initial posterior membership distributions of the data points are almost uniform and the Shannon entropy of the hidden random variables is very high. The change of this entropy over iterations for the News20 subsets is presented in Figure 6. The behavior is similar for all the other data sets that we studied. Unlike `kmeans`-based algorithms where most of the relocation happens in the first two or three iterations with only minor adjustments later on, in `soft-movMF` the data points are noncommittal in the first few iterations, and the entropy remains very high (the maximum possible entropy for 3 clusters can be $\log_2 3 = 1.585$). The cluster patterns are discovered only after several iterations, and the entropy drops drastically within a small number of iterations after that. When the algorithm converges, the entropy is practically zero and all points are effectively hard-assigned to their respective clusters. Note that this behavior is strikingly similar to (locally adaptive) annealing approaches where κ can be considered as the inverse of the temperature parameter. The drastic drop in entropy after a few iterations is the typical critical temperature behavior observed in annealing.

As text data has only non-negative features values, all the data points lie in the first orthant of the d -dimensional hypersphere and hence, are naturally very concentrated. The gene-expression data, though spread all over the hypersphere seemed to have some high concentration regions. In either case, the final κ values on convergence are very high, reflecting the concentration in the data, and implying a low final temperature from the annealing perspective. Then, initializing κ to a low value, or equivalently a high temperature, is a good idea because in that case when `soft-movMF` is running, the κ values will keep on increasing over successive iterations to get to its final large values, giving the effect of a decreasing temperature in the process, without any explicit deterministic annealing strategy. Also different mixture components can take different values of κ , as automatically determined by the EM algorithm itself, and need not be controlled by any external heuristic. The cost of the added flexibility in `soft-movMF` over `spkmeans` is the extra computation in estimating the κ values. Thus the `soft-movMF` algorithm provides a trade-off between modeling power and computational demands, but one that judging from the empirical results, seems quite worthwhile. The `hard-movMF` algorithm, instead of using the more general `vMF` model, suffers because of hard-assignments from the very beginning. The `fskmeans` and `spkmeans` do not do well for difficult data sets due to their hard assignment scheme as well as their significantly less modeling capabilities.

Finally, a word on model selection, since choosing the number of clusters remains one of the widely debated topics in clustering (McLachlan and Peel, 2000). Banerjee and Langford (2004) have proposed a new objective criterion for evaluation and model-selection for clustering algorithms: how well does the clustering algorithm perform as a prediction algorithm. The prediction

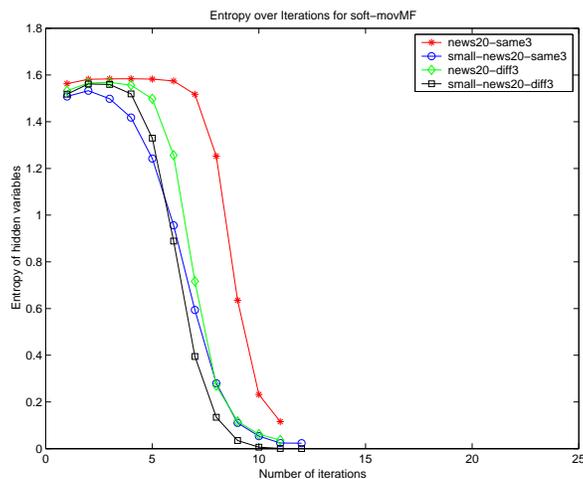


Figure 6: Variation of Entropy of hidden variables with number of Iterations (soft-movMF).

accuracy of the clustering is measured by the PAC-MDL bound (Blum and Langford, 2003; Banerjee and Langford, 2004) that upper-bounds the error-rate of predictions on the test-set. The way to use it for model-selection is quite straight-forward: among a range of number of clusters, choose the one that achieves the minimum bound on the test-set error-rate. Experiments on model selection applied to several clustering algorithms were reported by Banerjee and Langford (2004). Interestingly, the movMF-based algorithms almost always obtained the ‘right number of clusters’—in this case, the underlying labels in the data set were actually known and the number of labels were considered to be the right number of clusters. It is important to note that this form of model-selection only works in a semi-supervised setting where a little amount of labeled data is available for model selection.

8. Conclusions and Future Work

From the experimental results, it seems that certain high-dimensional data sets, including text and gene-expression data, have properties that match well with the modeling assumptions of the vMF mixture model. This motivates further study of such models. For example, one can consider a hybrid algorithm that employs soft-movMF for the first few (more important) iterations, and then switches to hard-movMF for speed, and measure the speed-quality tradeoff that this hybrid approach provides. Another possible extension would be to consider an online version of the EM-based algorithms as discussed in this paper, developed along the lines of Neal and Hinton (1998). Online algorithms are particularly attractive for dealing with streaming data when memory is limited, and for modeling mildly non-stationary data sources. We could also adapt a local search strategy such as the one in Dhillon et al. (2002a), for incremental EM to yield better local minima for both hard and soft-assignments.

The vMF distribution that we considered in the proposed techniques is one of the simplest parametric distributions for directional data. The iso-density lines of the vMF distribution are circles on the hypersphere, i.e., all points on the surface of the hypersphere at a constant angle from the mean direction. In some applications, more general iso-density contours may be desirable. There are

more general models on the unit sphere, such as the Bingham distribution, the Kent distribution, the Watson distribution, the Fisher-Bingham distribution, the Pearson type VII distributions (Shimizu and Iida, 2002; Mardia and Jupp, 2000), etc., that can potentially be more applicable in the general setting. For example, the Fisher-Bingham distributions have added modeling power since there are $O(d^2)$ parameters for each distribution. However, the parameter estimation problem, especially in high-dimensions, can be significantly more difficult for such models, as more parameters need to be estimated from the data. One definitely needs substantially more data to get reliable estimates of the parameters. Further, for some cases, e.g., the Kent distribution, it can be difficult to solve the estimation problem in more than 3-dimensions (Peel et al., 2001). Hence these more complex models may not be viable for many high-dimensional problems. Nevertheless, the tradeoff between model complexity (in terms of the number of parameters and their estimation), and sample complexity needs to be studied in more detail in the context of directional data.

Acknowledgments

This research was supported in part by an IBM PhD fellowship to Arindam Banerjee, NSF grant IIS-0307792, ITR-0312471, NSF CAREER Award No. ACI-0093404 and Texas Advanced Research Program grant 003658-0431-2001.

Appendix A. Derivations

For reference, we provide the derivation of Maximum Likelihood Estimates (MLE) for data drawn for a single vMF distribution (Section A.1), and Expectation Minimization update formulae for data drawn from a mixture of k vMF distributions (Section A.2).

A.1 Maximum Likelihood Estimates

In this section we look briefly at maximum likelihood estimates for the parameters of a single vMF distribution. Let \mathcal{X} be a finite set of sample unit vectors drawn independently following $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$ (see 2.1), i.e.,

$$\mathcal{X} = \{\mathbf{x}_i \in \mathbb{S}^{d-1} \mid \mathbf{x}_i \text{ follows } f(\mathbf{x}|\boldsymbol{\mu}, \kappa) \text{ for } 1 \leq i \leq n\}.$$

Given \mathcal{X} we want to find maximum likelihood estimates for the parameters $\boldsymbol{\mu}$ and κ of the distribution $f(\mathbf{x}|\boldsymbol{\mu}, \kappa)$. Assuming the \mathbf{x}_i to be independent and identically distributed, we can rewrite the likelihood of \mathcal{X} as

$$P(\mathcal{X}|\boldsymbol{\mu}, \kappa) = P(\mathbf{x}_1, \dots, \mathbf{x}_n|\boldsymbol{\mu}, \kappa) = \prod_{i=1}^n f(\mathbf{x}_i|\boldsymbol{\mu}, \kappa) = \prod_{i=1}^n c_d(\kappa) e^{\kappa \boldsymbol{\mu}^T \mathbf{x}_i}. \quad (\text{A.1})$$

Taking the logarithm on both sides of (A.1) we obtain

$$\ln P(\mathcal{X}|\boldsymbol{\mu}, \kappa) = n \ln c_d(\kappa) + \kappa \boldsymbol{\mu}^T \mathbf{r}, \quad (\text{A.2})$$

where $\mathbf{r} = \sum_i \mathbf{x}_i$. To obtain the maximum likelihood estimates of $\boldsymbol{\mu}$ and κ , we have to maximize (A.2), subject to the constraints $\boldsymbol{\mu}^T \boldsymbol{\mu} = 1$ and $\kappa \geq 0$. Introducing a Lagrange multiplier λ ,

the Lagrangian of the objective function is given by⁵

$$L(\boldsymbol{\mu}, \boldsymbol{\kappa}, \lambda; \mathcal{X}) = n \ln c_d(\boldsymbol{\kappa}) + \boldsymbol{\kappa} \boldsymbol{\mu}^T \mathbf{r} + \lambda(1 - \boldsymbol{\mu}^T \boldsymbol{\mu}). \quad (\text{A.3})$$

Differentiating the Lagrangian (A.3) with respect to $\boldsymbol{\mu}$, λ and $\boldsymbol{\kappa}$ and setting the derivatives to zero, we get the following equations that the parameter estimates $\hat{\boldsymbol{\mu}}$, $\hat{\lambda}$ and $\hat{\boldsymbol{\kappa}}$ must satisfy:

$$\hat{\boldsymbol{\mu}} = \frac{\hat{\boldsymbol{\kappa}}}{2\hat{\lambda}} \mathbf{r}, \quad (\text{A.4a})$$

$$\hat{\boldsymbol{\mu}}^T \hat{\boldsymbol{\mu}} = 1, \quad (\text{A.4b})$$

$$\frac{nc'_d(\hat{\boldsymbol{\kappa}})}{c_d(\hat{\boldsymbol{\kappa}})} = -\hat{\boldsymbol{\mu}}^T \mathbf{r}. \quad (\text{A.4c})$$

Substituting (A.4a) in (A.4b) gives us

$$\hat{\lambda} = \frac{\hat{\boldsymbol{\kappa}}}{2} \|\mathbf{r}\|, \quad (\text{A.5})$$

$$\text{and } \hat{\boldsymbol{\mu}} = \frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{\sum_{i=1}^n \mathbf{x}_i}{\|\sum_{i=1}^n \mathbf{x}_i\|}. \quad (\text{A.6})$$

Substituting $\hat{\boldsymbol{\mu}}$ from (A.6) in (A.4c) we obtain

$$\frac{c'_d(\hat{\boldsymbol{\kappa}})}{c_d(\hat{\boldsymbol{\kappa}})} = -\frac{\|\mathbf{r}\|}{n} = -\bar{r}. \quad (\text{A.7})$$

For brevity, let us write $s = d/2 - 1$ and $\xi = (2\pi)^{s+1}$; on differentiating (2.2) with respect to $\boldsymbol{\kappa}$, we obtain

$$c'_d(\boldsymbol{\kappa}) = \frac{s\boldsymbol{\kappa}^{s-1}}{\xi I_s(\boldsymbol{\kappa})} - \frac{\boldsymbol{\kappa}^s I'_s(\boldsymbol{\kappa})}{\xi I_s^2(\boldsymbol{\kappa})}.$$

The right-hand-side simplifies to

$$\frac{\boldsymbol{\kappa}^s}{\xi I_s(\boldsymbol{\kappa})} \left(\frac{s}{\boldsymbol{\kappa}} - \frac{I'_s(\boldsymbol{\kappa})}{I_s(\boldsymbol{\kappa})} \right) = c_d(\boldsymbol{\kappa}) \left(\frac{s}{\boldsymbol{\kappa}} - \frac{I'_s(\boldsymbol{\kappa})}{I_s(\boldsymbol{\kappa})} \right).$$

Using the following well known recurrence relation (see Abramowitz and Stegun (1974, Sec. 9.6.26)),

$$\boldsymbol{\kappa} I_{s+1}(\boldsymbol{\kappa}) = \boldsymbol{\kappa} I'_s(\boldsymbol{\kappa}) - s I_s(\boldsymbol{\kappa}),$$

we find that

$$\frac{-c'_d(\boldsymbol{\kappa})}{c_d(\boldsymbol{\kappa})} = \frac{I_{s+1}(\boldsymbol{\kappa})}{I_s(\boldsymbol{\kappa})} = \frac{I_{d/2}(\boldsymbol{\kappa})}{I_{d/2-1}(\boldsymbol{\kappa})}.$$

Thus we can obtain the estimate $\hat{\boldsymbol{\kappa}}$ by solving

$$A_d(\hat{\boldsymbol{\kappa}}) = \bar{r}, \quad (\text{A.8})$$

where $A_d(\boldsymbol{\kappa}) = \frac{I_{d/2}(\boldsymbol{\kappa})}{I_{d/2-1}(\boldsymbol{\kappa})}$ and $\bar{r} = \|\mathbf{r}\|/n$. Since $A_d(\boldsymbol{\kappa})$ is a ratio of Bessel functions, it is not possible to obtain a closed form expression for A_d^{-1} . We have to take recourse to numerical or asymptotic methods to obtain an approximation for $\boldsymbol{\kappa}$.

5. Strictly speaking, we should introduce the inequality constraint in the Lagrangian for $\boldsymbol{\kappa}$, and work with the necessary KKT conditions. However if $\boldsymbol{\kappa} = 0$ then $f(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\kappa})$ is the uniform distribution on the sphere, and if $\boldsymbol{\kappa} > 0$ then the multiplier for the inequality constraint has to be zero by the KKT condition, so the Lagrangian in (A.3) is adequate.

A.2 Expectation Maximization (EM)

Suppose the posterior distribution, $p(h|\mathbf{x}_i, \Theta)$, of the hidden variables $Z|(\mathcal{X}, \Theta)$ is known. Unless otherwise specified, henceforth all expectations will be taken over the distribution of the (set of) random variable(s) $Z|(\mathcal{X}, \Theta)$. Expectation of the complete data log-likelihood (see 3.2) over the given posterior distribution p can be written as

$$\begin{aligned} E_p[\ln P(\mathcal{X}, Z|\Theta)] &= \sum_{i=1}^n E_p[\ln(\alpha_{z_i} f_{z_i}(\mathbf{x}_i|\theta_{z_i}))] \\ &= \sum_{i=1}^n \sum_{h=1}^k \ln(\alpha_h f_h(\mathbf{x}_i|\theta_h)) p(h|\mathbf{x}_i, \Theta) \\ &= \sum_{h=1}^k \sum_{i=1}^n (\ln \alpha_h) p(h|\mathbf{x}_i, \Theta) + \sum_{h=1}^k \sum_{i=1}^n (\ln f_h(\mathbf{x}_i|\theta_h)) p(h|\mathbf{x}_i, \Theta). \end{aligned} \quad (\text{A.9})$$

In the parameter estimation or M-step, Θ is re-estimated so that the above expression is maximized. Note that for maximizing this expectation we can separately maximize the terms containing α_h and θ_h as they are unrelated (observe that $p(h|\mathbf{x}_i, \Theta)$ is fixed).

To maximize the expectation with respect to each α_h we introduce a Lagrangian multiplier λ corresponding to the constraint $\sum_{h=1}^k \alpha_h = 1$. We form the Lagrangian, and take partial derivatives with respect to each α_h obtaining

$$\sum_{i=1}^n p(h|\mathbf{x}_i, \Theta) = -\lambda \hat{\alpha}_h. \quad (\text{A.10})$$

On summing both sides of (A.10) over all h we find that $\lambda = -n$, hence

$$\hat{\alpha}_h = \frac{1}{n} \sum_{i=1}^n p(h|\mathbf{x}_i, \Theta). \quad (\text{A.11})$$

Next we concentrate on terms containing $\theta_h = (\boldsymbol{\mu}_h, \kappa_h)$ under the constraints $\boldsymbol{\mu}_h^T \boldsymbol{\mu}_h = 1$ and $\kappa_h \geq 0$ for $1 \leq h \leq k$. Let λ_h be the Lagrange multiplier corresponding to each equality constraint (see footnote on page 1374). The Lagrangian is given by

$$\begin{aligned} L(\{\boldsymbol{\mu}_h, \kappa_h, \lambda_h\}_{h=1}^k) &= \sum_{h=1}^k \sum_{i=1}^n (\ln f_h(\mathbf{x}_i|\theta_h)) p(h|\mathbf{x}_i, \Theta) + \sum_{h=1}^k \lambda_h (1 - \boldsymbol{\mu}_h^T \boldsymbol{\mu}_h) \\ &= \sum_{h=1}^k \left[\sum_{i=1}^n (\ln c_d(\kappa_h)) p(h|\mathbf{x}_i, \Theta) + \sum_{i=1}^n \kappa_h \boldsymbol{\mu}_h^T \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) + \lambda_h (1 - \boldsymbol{\mu}_h^T \boldsymbol{\mu}_h) \right]. \end{aligned} \quad (\text{A.12})$$

Taking partial derivatives of (A.12) with respect to $\{\boldsymbol{\mu}_h, \lambda_h, \kappa_h\}_{h=1}^k$ and setting them to zero, for each h we get:

$$\boldsymbol{\mu}_h = \frac{\kappa_h}{2\lambda_h} \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta), \quad (\text{A.13a})$$

$$\boldsymbol{\mu}_h^T \boldsymbol{\mu}_h = 1, \quad (\text{A.13b})$$

$$\frac{c'_d(\kappa_h)}{c_d(\kappa_h)} \sum_{i=1}^n p(h|\mathbf{x}_i, \Theta) = -\boldsymbol{\mu}_h^T \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta). \quad (\text{A.13c})$$

Using (A.13a) and (A.13b) we get

$$\begin{aligned}\lambda_h &= \frac{\kappa_h}{2} \left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|, \\ \boldsymbol{\mu}_h &= \frac{\sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta)}{\left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|}.\end{aligned}\tag{A.14}$$

Substituting (A.14) in (A.13c) gives us

$$\frac{c'_d(\kappa_h)}{c_d(\kappa_h)} = - \frac{\left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|}{\sum_{i=1}^n p(h|\mathbf{x}_i, \Theta)},\tag{A.15}$$

which can be written as

$$A_d(\kappa_h) = \frac{\left\| \sum_{i=1}^n \mathbf{x}_i p(h|\mathbf{x}_i, \Theta) \right\|}{\sum_{i=1}^n p(h|\mathbf{x}_i, \Theta)},\tag{A.16}$$

where $A_d(\kappa) = \frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}$. Note that (A.14) and (A.16) are intuitive generalizations of (A.6) and (A.8) respectively.

A.3 Experimental Study of the Approximation

In this section we provide a brief experimental study to assess the quality of our approximation of the concentration parameter κ . Recall that our approximation (4.4) attempts to solve the implicit non-linear equation

$$\frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)} = \bar{r}.\tag{A.17}$$

We previously mentioned that for large values of \bar{r} (\bar{r} close to 1), approximation (4.1) is reasonable; for small values of \bar{r} (usually for $\bar{r} < 0.2$) estimate (4.2) is quite good; Eqn. (4.4) yields good approximations for most values of \bar{r} .

A particular value of \bar{r} may correspond to many different combinations of κ and d values. Thus, to assess the quality of various approximations, we need to evaluate their performance across the (κ, d) plane. However, such an assessment is difficult to illustrate through 2-dimensional plots. To supplement Table 1, which showed how the three approximations behave on a sampling of points from the (κ, d) plane, in this section we present experimental results on some slices of this plane, where we either keep d fixed and vary κ , or we keep κ fixed and vary d . For all our evaluations, the \bar{r} values were computed using (A.17).

We begin by holding d fixed at 1000, and allow κ to vary from 10 to 5010. Figure 7 shows the values of computed $\hat{\kappa}$ (estimation of κ) using the three approximations. From this figure one can see that (4.1) overestimates the true κ , while (4.2) underestimates it. However, our approximation (4.4) is very close to the true κ values.

Next we illustrate the quality of approximation when κ is held fixed and d is allowed to vary. Figure 8 illustrates how the various approximations behave as the dimensionality d is varied from $d = 4$ till $d = 1454$. The concentration parameter κ was set at 500 for this experiment. We see that (4.2) catches up with the true value of κ after approximately $d \geq 2\kappa$ (because the associated \bar{r} values become small), whereas (4.4) remains accurate throughout.

Since all the approximations depend on \bar{r} (which implicitly depends on κ and d), it is illustrative to also plot the approximation errors as \bar{r} is allowed to vary. Figure 9 shows how the three

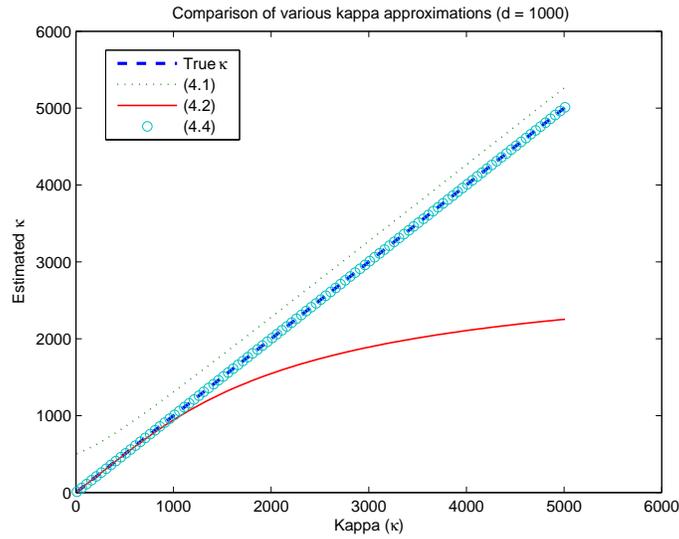


Figure 7: Comparison of true and approximated κ values, with $d = 1000$

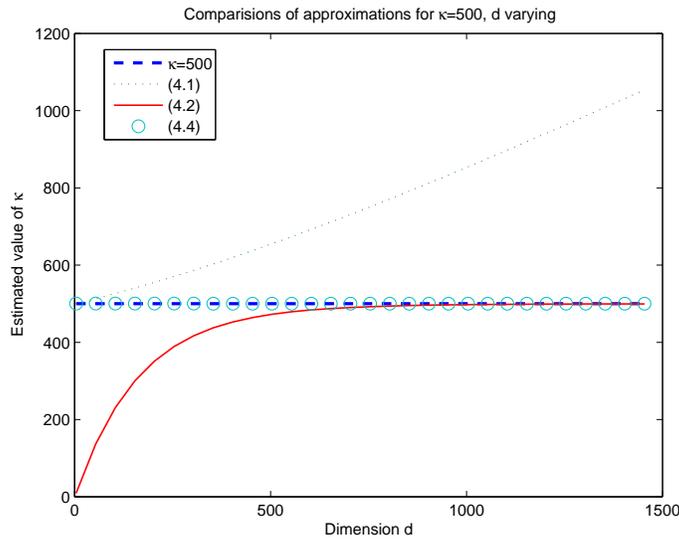


Figure 8: Comparison of approximations for varying d , $\kappa = 500$.

approximations perform as \bar{r} ranges from 0.05 to 0.95. Let $f(d, \bar{r})$, $g(d, \bar{r})$, and $h(d, \bar{r})$ represent the approximations to κ using (4.1), (4.2) and (4.4), respectively. Figure 9 displays $|A_d(f(d, \bar{r})) - \bar{r}|$, $|A_d(g(d, \bar{r})) - \bar{r}|$, and $|A_d(h(d, \bar{r})) - \bar{r}|$ for the varying \bar{r} values. Note that the y-axis is on a log-scale to appreciate the differences between the three approximations. We see that up to $\bar{r} \approx 0.18$ (dashed line on the plot), the approximation yielded by (4.2) has lower error. Thereafter, approximation (4.4) becomes better.

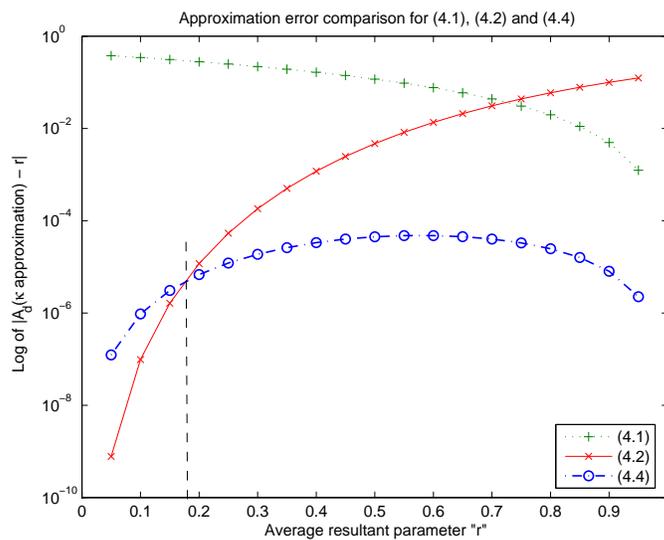


Figure 9: Comparison of approximations for varying \bar{r} (with $d = 1000$)

References

- M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions*. Dover Publ. Inc., New York, 1974.
- S. I. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995.
- A. Banerjee and J. Ghosh. Frequency sensitive competitive learning for clustering on high-dimensional hyperspheres. In *Proceedings International Joint Conference on Neural Networks*, pages 1590–1595, May 2002.
- A. Banerjee and J. Ghosh. Frequency Sensitive Competitive Learning for Scalable Balanced Clustering on High-dimensional Hyperspheres. *IEEE Transactions on Neural Networks*, 15(3):702–719, May 2004.
- A. Banerjee and J. Langford. An objective evaluation criterion for clustering. In *Proc. 10th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 515–520, 2004.
- A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. In *Proc. of 4th SIAM International Conference on Data Mining*, pages 234–245, 2004.
- J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical Report ICSI-TR-97-021, University of Berkeley, 1997.
- A. Blum and J. Langford. PAC-MDL bounds. In *Proc. 16th Annual Conference on Learning Theory (COLT)*, 2003.

- P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical report, Microsoft Research, May 2000.
- D. Coleman, X. Dong, J. Hardin, D. Rocke, and D. Woodruff. Some computational issues in cluster analysis with no a priori metric. *Computational Statistics and Data Analysis*, 31:1–11, 1999.
- M. Collins. The EM algorithm. In fulfillment of Written Preliminary Exam II requirement, September 1997.
- T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.
- S. Dasgupta. Learning mixtures of Gaussians. In *IEEE Symposium on Foundations of Computer Science*, 1999.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.
- I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. Kumar, R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, 2001.
- I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of The 2002 IEEE International Conference on Data Mining*, 2002a.
- I. S. Dhillon, E. M. Marcotte, and U. Roshan. Diametrical clustering for identifying anti-correlated gene clusters. Technical Report TR 2002-49, Department of Computer Sciences, University of Texas, September 2002b.
- I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.
- I. S. Dhillon and S. Sra. Modeling data using directional distributions. Technical Report TR-03-06, Department of Computer Sciences, University of Texas at Austin, Austin, TX, 2003.
- B. E. Dom. An information-theoretic external cluster-validity measure. Technical Report RJ 10219, IBM Research Report, 2001.
- M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. National Academy of Science*, 95:14863–14868, 1998.
- N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1996.
- J. Ghosh. Scalable clustering methods for data mining. In Nong Ye, editor, *Handbook of Data Mining*, pages 247–277. Lawrence Erlbaum, 2003.
- GoMiner03. <http://discover.nci.nih.gov/gominer/>.
- T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, D. D. Shoemaker, D. Gachotte, K. Chakraborty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

- P. Indyk. A sublinear-time approximation scheme for clustering in metric spaces. In *40th Symposium on Foundations of Computer Science*, 1999.
- T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 487–493. MIT Press, 1999.
- A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.
- R. Kannan, S. Vempala, and A. Vetta. On clusterings—good, bad and spectral. In *41st Annual IEEE Symposium Foundations of Computer Science*, pages 367–377, 2000.
- M. Kearns, Y. Mansour, and A. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *13th Annual Conf. on Uncertainty in Artificial Intelligence (UAI97)*, 1997.
- I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306(5701):1555–1558, 2004.
- K. V. Mardia. *Statistical Distributions in Scientific Work*, volume 3, chapter “Characteristics of directional distributions”, pages 365–385. Reidel, Dordrecht, 1975.
- K. V. Mardia and P. Jupp. *Directional Statistics*. John Wiley and Sons Ltd., 2nd edition, 2000.
- G. J. McLachlan. The classification and mixture maximum likelihood approaches to cluster analysis. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, volume 2, pages 199–208. North-Holland, 1982.
- G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 1997.
- G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley series in Probability and Mathematical Statistics: Applied Probability and Statistics Section. John Wiley & Sons, 2000.
- M. Meilă. Comparing clusterings by the variation of information. In *COLT*, 2003.
- J. A. Mooney, P. J. Helms, and I. T. Jolliffe. Fitting mixtures of von Mises distributions: a case study involving sudden infant death syndrome. *Computational Statistics & Data Analysis*, 41: 505–513, 2003.
- R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.
- 20 Newsgroups. <http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html>.
- K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.
- D. Peel, W. J. Whiten, and G. J. McLachlan. Fitting mixtures of Kent distributions to aid in joint set identification. *Journal of American Statistical Association*, 96:56–63, 2001.

- J. H. Piater. *Visual Feature Learning*. PhD thesis, University of Massachusetts, June 2001.
- C. R. Rao. *Linear Statistical Inference and its Applications*. Wiley, New York, 2nd edition, 1973.
- E. Rasmussen. Clustering algorithms. In W. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, New Jersey, 1992.
- G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):513–523, 1988.
- G. Salton and M. J. McGill. *Introduction to Modern Retrieval*. McGraw-Hill Book Company, 1983.
- B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, 10, pages 285–295, 2001.
- B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.
- E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *Proc. of 8th Pacific Symposium on Biocomputing (PSB)*, 2003.
- R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Proceedings of 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 307–316. AAAI Press, 2000.
- K. Shimizu and K. Iida. Pearson type VII distributions on spheres. *Communications in Statistics: Theory & Methods*, 31(4):513–526, 2002.
- J. Sinkkonen and S. Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239, 2001.
- P. Smyth. Clustering sequences with hidden Markov models. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing*, volume 9, pages 648–654. MIT Press, 1997.
- M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
- A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc 7th Natl Conf on Artificial Intelligence : Workshop of AI for Web Search (AAAI 2000)*, pages 58–64. AAAI, July 2000.
- C. S. Wallace and D. L. Dowe. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10(1):73–83, January 2000.
- G. N. Watson. *A treatise on the theory of Bessel functions*. Cambridge University Press, 2nd edition, 1995.
- A. T. A. Wood. Simulation of the von-Mises Distribution. *Communications of Statistics, Simulation and Computation*, 23:157–164, 1994.
- Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, June 2004.

- S. Zhong and J. Ghosh. A Unified Framework for Model-based Clustering. *Journal of Machine Learning Research*, 4:1001–1037, November 2003a.
- S. Zhong and J. Ghosh. A comparative study of generative models for document clustering. In *Workshop on Clustering High Dimensional Data : Third SIAM Conference on Data Mining*, April 2003b.

Inner Product Spaces for Bayesian Networks

Atsuyoshi Nakamura

*Graduate School of Information Science and Technology
Hokkaido University
Sapporo 060-8628, Japan*

ATSU@MAIN.IST.HOKUDAI.AC.JP

Michael Schmitt

Niels Schmitt

Hans Ulrich Simon

*Lehrstuhl Mathematik und Informatik
Fakultät für Mathematik
Ruhr-Universität Bochum
44780 Bochum, Germany*

MSCHMITT@LMI.RUHR-UNI-BOCHUM.DE

NSCHMITT@LMI.RUHR-UNI-BOCHUM.DE

SIMON@LMI.RUHR-UNI-BOCHUM.DE

Editor: Tommi Jaakkola

Abstract

Bayesian networks have become one of the major models used for statistical inference. We study the question whether the decisions computed by a Bayesian network can be represented within a low-dimensional inner product space. We focus on two-label classification tasks over the Boolean domain. As main results we establish upper and lower bounds on the dimension of the inner product space for Bayesian networks with an explicitly given (full or reduced) parameter collection. In particular, these bounds are tight up to a factor of 2. For some nontrivial cases of Bayesian networks we even determine the exact values of this dimension. We further consider logistic autoregressive Bayesian networks and show that every sufficiently expressive inner product space must have dimension at least $\Omega(n^2)$, where n is the number of network nodes. We also derive the bound $2^{\Omega(n)}$ for an artificial variant of this network, thereby demonstrating the limits of our approach and raising an interesting open question. As a major technical contribution, this work reveals combinatorial and algebraic structures within Bayesian networks such that known methods for the derivation of lower bounds on the dimension of inner product spaces can be brought into play.

Keywords: Bayesian network, inner product space, embedding, linear arrangement, Euclidean dimension

1. Introduction

During the last decade, there has been remarkable interest in learning systems based on hypotheses that can be written as inner products in an appropriate feature space and learned by algorithms that perform a kind of empirical or structural risk minimization. Often in such systems the inner product operation is not carried out explicitly, but reduced to the evaluation of a so-called kernel function that operates on instances of the original data space. A major advantage of this technique is that it allows to handle high-dimensional feature spaces efficiently. The learning strategy proposed by Boser et al. (1992) in connection with the so-called support vector machine is a theoretically well founded and very powerful method that, in the years since its introduction, has already outperformed most other systems in a wide variety of applications (see also Vapnik, 1998).

Bayesian networks have a long history in statistics. In the first half of the 1980s they were introduced to the field of expert systems through work by Pearl (1982) and Spiegelhalter and Knill-Jones (1984). Bayesian networks are much different from kernel-based learning systems and offer some complementary advantages. They graphically model conditional independence relationships between random variables. Like other probabilistic models, Bayesian networks can be used to represent inhomogeneous data with possibly overlapping features and missing values in a uniform manner. Quite elaborate methods dealing with Bayesian networks have been developed for solving problems in pattern classification.

One of the motivations for the work this article is about was that recently several research groups considered the possibility of combining the key advantages of probabilistic models and kernel-based learning systems. Various kernels were suggested and extensively studied, for instance, by Jaakkola and Haussler (1999a,b), Oliver et al. (2000), Saunders et al. (2003), Tsuda and Kawanabe (2002), and Tsuda et al. (2002, 2004). Altun et al. (2003) proposed a kernel for the Hidden Markov Model, which is a special case of a Bayesian network. Another approach for combining kernel methods and probabilistic models has been made by Taskar et al. (2004).

In this article, we consider Bayesian networks as computational models that perform two-label classification tasks over the Boolean domain. We aim at finding the simplest inner product space that is able to express the concept class, that is, the class of decision functions, induced by a given Bayesian network. Hereby, “simplest” refers to a space which has as few dimensions as possible. We focus on Euclidean spaces equipped with the standard dot product. For the finite-dimensional case, this is no loss of generality since any finite-dimensional reproducing kernel Hilbert space is isometric with \mathbb{R}^d for some d . Furthermore, we use the Euclidean dimension of the space as the measure of complexity. This is well motivated by the fact that most generalization error bounds for linear classifiers are given in terms of either the Euclidean dimension or in terms of the geometrical margin between the data points and the separating hyperplanes. Applying random projection techniques from Johnson and Lindenstrauss (1984), Frankl and Maehara (1988), or Arriaga and Vempala (1999), it can be shown that any arrangement with a large margin can be converted into a low-dimensional arrangement. A recent result of Balcan et al. (2004) in this direction even takes into account low-dimensional arrangements that allow a certain amount of error. Thus, a large lower bound on the smallest possible dimension rules out the possibility that a classifier with a large margin exists. Given a Bayesian network \mathcal{N} , we introduce $\text{Edim}(\mathcal{N})$ for denoting the smallest dimension d such that the decisions represented by \mathcal{N} can be implemented as inner products in the d -dimensional Euclidean space. Our results are provided as upper and lower bounds for $\text{Edim}(\mathcal{N})$.

We first consider Bayesian networks with an explicitly given parameter collection. The parameters can be arbitrary, where we speak of an unconstrained network, or they may be required to satisfy certain restrictions, in which case we have a network with a reduced parameter collection. For both network types, we show that the “natural” inner product space, which can be obtained from the probabilistic model by straightforward algebraic manipulations, has a dimension that is the smallest possible up to a factor of 2, and even up to an additive term of 1 in some cases. Furthermore, we determine the exact values of $\text{Edim}(\mathcal{N})$ for some nontrivial instances of these networks. The lower bounds in all these cases are obtained by analyzing the Vapnik-Chervonenkis (VC) dimension of the concept class associated with the Bayesian network. Interestingly, the VC dimension plays also a major role when estimating the sample complexity of a learning system. In particular, it can be used to derive bounds on the number of training examples that are required for selecting hypotheses that generalize well on new data. Thus, the tight bounds on $\text{Edim}(\mathcal{N})$ reveal that the smallest possible

Euclidean dimension for a Bayesian network with an explicitly given parameter collection is closely tied to its sample complexity.

As a second topic, we investigate a class of probabilistic models known as logistic autoregressive Bayesian networks or sigmoid belief networks. These networks were originally proposed by McCullagh and Nelder (1983) and studied systematically, for instance, by Neal (1992), and Saul et al. (1996). (See also Frey, 1998). Using the VC dimension, we show that $\text{Edim}(\mathcal{N})$ for these networks must grow at least as $\Omega(n^2)$, where n is the number of nodes.

Finally, we get interested in the question whether it is possible to establish an exponential lower bound on $\text{Edim}(\mathcal{N})$ for the logistic autoregressive Bayesian network. This investigation is motivated by the fact we also derive here that these networks have their VC dimension bounded by $O(n^6)$. Consequently, VC dimension considerations are not sufficient to yield an exponential lower bound for $\text{Edim}(\mathcal{N})$. We succeed in giving a positive answer for an unnatural variant of this network that we introduce and call the modified logistic autoregressive Bayesian network. This variant is also shown to have VC dimension $O(n^6)$. We obtain that for a network with $n + 2$ nodes, $\text{Edim}(\mathcal{N})$ is at least as large as $2^{n/4}$. The proof for this lower bound is based on the idea of embedding one concept class into another. In particular, we show that a certain class of Boolean parity functions can be embedded into such a network.

While, as mentioned above, the connection between probabilistic models and inner product spaces has already been investigated, this work seems to be the first one that explicitly addresses the question of finding a smallest-dimensional sufficiently expressive inner product space. In addition, there has been related research considering the question of representing a given concept class by a system of halfspaces, but not concerned with probabilistic models (see, e.g., Ben-David et al., 2002; Forster et al., 2001; Forster, 2002; Forster and Simon, 2002; Forster et al., 2003; Kiltz, 2003; Kiltz and Simon, 2003; Srebro and Shraibman, 2005; Warmuth and Vishwanathan, 2005). A further contribution of our work can be seen in the uncovering of combinatorial and algebraic structures within Bayesian networks such that techniques known from this literature can be brought into play.

We start by introducing the basic concepts in Section 2. The upper bounds are presented in Section 3. Section 4 deals with lower bounds that are obtained using the VC dimension as the core tool. The exponential lower bound for the modified logistic autoregressive network is derived in Section 5. In Section 6 we draw the major conclusions and mention some open problems.

Bibliographic Note. Results in this article have been presented at the 17th Annual Conference on Learning Theory, COLT 2004, in Banff, Canada (Nakamura et al., 2004).

2. Preliminaries

In the following, we give formal definitions for the basic notions in this article. Section 2.1 introduces terminology from learning theory. In Section 2.2, we define Bayesian networks and the distributions and concept classes they induce. The idea of a linear arrangement for a concept class is presented in Section 2.3.

2.1 Concept Classes, VC Dimension, and Embeddings

A *concept class* \mathcal{C} over domain X is a family of functions of the form $f : X \rightarrow \{-1, 1\}$. Each $f \in \mathcal{C}$ is called a *concept*. A finite set $S = \{s_1, \dots, s_m\} \subseteq X$ is said to be *shattered* by \mathcal{C} if for every binary vector $b \in \{-1, 1\}^m$ there exists some concept $f \in \mathcal{C}$ such that $f(s_i) = b_i$ for $i = 1, \dots, m$. The

Vapnik-Chervonenkis (VC) dimension of \mathcal{C} is given by

$$\text{VCdim}(\mathcal{C}) = \sup\{m \mid \text{there is some } S \subseteq \mathcal{X} \text{ shattered by } \mathcal{C} \text{ and } |S| = m\}.$$

For every $z \in \mathbb{R}$, let $\text{sign}(z) = 1$ if $z \geq 0$, and $\text{sign}(z) = -1$ otherwise. We use the sign function for mapping a real-valued function g to a ± 1 -valued concept $\text{sign} \circ g$.

Given a concept class \mathcal{C} over domain \mathcal{X} and a concept class \mathcal{C}' over domain \mathcal{X}' , we write $\mathcal{C} \leq \mathcal{C}'$ if there exist mappings

$$\mathcal{C} \ni f \mapsto f' \in \mathcal{C}' \quad \text{and} \quad \mathcal{X} \ni x \mapsto x' \in \mathcal{X}'$$

satisfying

$$f(x) = f'(x') \text{ for every } f \in \mathcal{C} \text{ and } x \in \mathcal{X}.$$

These mappings are said to provide an *embedding* of \mathcal{C} into \mathcal{C}' . Obviously, if $S \subseteq \mathcal{X}$ is an m -element set that is shattered by \mathcal{C} then $S' = \{s' \mid s \in S\} \subseteq \mathcal{X}'$ is an m -element set that is shattered by \mathcal{C}' . Consequently, $\mathcal{C} \leq \mathcal{C}'$ implies $\text{VCdim}(\mathcal{C}) \leq \text{VCdim}(\mathcal{C}')$.

2.2 Bayesian Networks

Definition 1 A Bayesian network \mathcal{N} has the following components:

1. A directed acyclic graph $G = (V, E)$, where V is a finite set of nodes and $E \subseteq V \times V$ a set of edges,
2. a collection $(p_{i,\alpha})_{i \in V, \alpha \in \{0,1\}^{m_i}}$ of programmable parameters with values in the open interval $]0, 1[$, where m_i denotes the number of predecessors of node i , that is, $m_i = |\{j \in V \mid (j, i) \in E\}|$,
3. constraints that describe which assignments of values from $]0, 1[$ to the parameters of the collection are allowed.

If the constraints are empty, we speak of an *unconstrained network*. Otherwise, the network is constrained.

We identify the $n = |V|$ nodes of \mathcal{N} with the numbers $1, \dots, n$ and assume that every edge $(j, i) \in E$ satisfies $j < i$, that is, E induces a topological ordering on $\{1, \dots, n\}$. Given $(j, i) \in E$, j is called a parent of i . We use P_i to denote the set of parents of node i , and let $m_i = |P_i|$ be the number of parents. A network \mathcal{N} is said to be *fully connected* if $P_i = \{1, \dots, i - 1\}$ holds for every node i .

Example 1 (kth-order Markov chain) For $k \geq 0$, let \mathcal{N}_k denote the unconstrained Bayesian network with $P_i = \{i - 1, \dots, i - k\}$ for $i = 1, \dots, n$ (with the convention that numbers smaller than 1 are ignored such that $m_i = |P_i| = \min\{i - 1, k\}$). The total number of parameters is equal to $2^k(n - k) + 2^{k-1} + \dots + 2 + 1 = 2^k(n - k + 1) - 1$.

We associate with every node i a Boolean variable x_i with values in $\{0, 1\}$. We say x_j is a parent-variable of x_i if j is a parent of i . Each $\alpha \in \{0, 1\}^{m_i}$ is called a possible bit-pattern for the parent-variables of x_i . We use $M_{i,\alpha}$ to denote the polynomial

$$M_{i,\alpha}(x) = \prod_{j \in P_i} x_j^{\alpha_j}, \text{ where } x_j^0 = 1 - x_j \text{ and } x_j^1 = x_j,$$

that is, $M_{i,\alpha}(x)$ is 1 if the parent variables of x_i exhibit bit-pattern α , otherwise it is 0.

Bayesian networks are graphical models of conditional independence relationships. This general idea is made concrete by the following notion.

Definition 2 *Let \mathcal{N} be a Bayesian network with nodes $1, \dots, n$. The class of distributions induced by \mathcal{N} , denoted as $\mathcal{D}_{\mathcal{N}}$, consists of all distributions on $\{0, 1\}^n$ of the form*

$$P(x) = \prod_{i=1}^n \prod_{\alpha \in \{0,1\}^{m_i}} p_{i,\alpha}^{x_i M_{i,\alpha}(x)} (1 - p_{i,\alpha})^{(1-x_i) M_{i,\alpha}(x)}. \quad (1)$$

Thus, for every assignment of values from $]0, 1[$ to the parameters of \mathcal{N} , we obtain a specific distribution from $\mathcal{D}_{\mathcal{N}}$. Recall that not every possible assignment is allowed if \mathcal{N} is constrained.

The polynomial representation of $\log(P(x))$ resulting from equation (1) is known as ‘‘Chow expansion’’ in the pattern classification literature (see, e.g., Duda and Hart, 1973). The parameter $p_{i,\alpha}$ represents the conditional probability for the event $x_i = 1$ given that the parent variables of x_i exhibit bit-pattern α . Equation (1) is a chain expansion for $P(x)$ that expresses $P(x)$ as a product of conditional probabilities.

An unconstrained network that is highly connected may have a number of parameters that grows exponentially in the number of nodes. The idea of a constrained network is to keep the number of parameters reasonably small even in case of a dense topology. We consider two types of constraints giving rise to the definitions of networks with a reduced parameter collection and logistic autoregressive networks.

Definition 3 *A Bayesian network with a reduced parameter collection is a Bayesian network with the following constraints: For every $i \in \{1, \dots, n\}$ there exists a surjective function $R_i : \{0, 1\}^{m_i} \rightarrow \{1, \dots, d_i\}$ such that the parameters of \mathcal{N} satisfy*

$$\forall i = 1, \dots, n, \forall \alpha, \alpha' \in \{0, 1\}^{m_i} : R_i(\alpha) = R_i(\alpha') \implies p_{i,\alpha} = p_{i,\alpha'}.$$

We denote the network as \mathcal{N}^R for $R = (R_1, \dots, R_n)$. Obviously, \mathcal{N}^R is completely described by the reduced parameter collection $(p_{i,c})_{1 \leq i \leq n, 1 \leq c \leq d_i}$.

A special case of these networks uses decision trees or graphs to represent the parameters.

Example 2 *Chickering et al. (1997) proposed Bayesian networks ‘‘with local structure’’. These networks contain a decision tree T_i (or, alternatively, a decision graph G_i) over the parent-variables of x_i for every node i . The conditional probability for $x_i = 1$, given the bit-pattern of the variables from P_i , is attached to the corresponding leaf in T_i (or sink in G_i , respectively). This fits nicely into our framework of networks with a reduced parameter collection. Here, d_i denotes the number of leaves in T_i (or sinks of G_i , respectively), and $R_i(\alpha)$ is equal to $c \in \{1, \dots, d_i\}$ if α is routed to leaf c in T_i (or to sink c in G_i , respectively).*

For a Bayesian network with reduced parameter collection, the distribution $P(x)$ from Definition 2 can be written in a simpler way. Let $R_{i,c}(x)$ denote the $\{0, 1\}$ -valued function that indicates for every $x \in \{0, 1\}^n$ whether the projection of x to the parent-variables of x_i is mapped by R_i to the value c . Then, we have

$$P(x) = \prod_{i=1}^n \prod_{c=1}^{d_i} p_{i,c}^{x_i R_{i,c}(x)} (1 - p_{i,c})^{(1-x_i) R_{i,c}(x)}. \quad (2)$$

We finally introduce the so-called logistic autoregressive Bayesian networks, originally proposed by McCullagh and Nelder (1983), that have been shown to perform surprisingly well on certain problems (see also Neal, 1992, Saul et al., 1996, and Frey, 1998).

Definition 4 *The logistic autoregressive Bayesian network $\mathcal{N}_{\mathcal{G}}$ is the fully connected Bayesian network with constraints on the parameter collection given as*

$$\forall i = 1, \dots, n, \exists (w_{i,j})_{1 \leq j \leq i-1} \in \mathbb{R}^{i-1}, \forall \alpha \in \{0, 1\}^{i-1} : p_{i,\alpha} = \sigma \left(\sum_{j=1}^{i-1} w_{i,j} \alpha_j \right),$$

where $\sigma(y) = 1/(1 + e^{-y})$ is the standard sigmoid function. Obviously, $\mathcal{N}_{\mathcal{G}}$ is completely described by the parameter collection $(w_{i,j})_{1 \leq i \leq n, 1 \leq j \leq i-1}$.

In a two-label classification task, functions $P(x), Q(x) \in \mathcal{D}_{\mathcal{N}}$ are used as discriminant functions, where $P(x)$ and $Q(x)$ represent the distributions of x conditioned to label 1 and -1 , respectively. The corresponding decision function assigns label 1 to x if $P(x) \geq Q(x)$, and -1 otherwise. The obvious connection to concept classes in learning theory is made explicit in the following definition.

Definition 5 *Let \mathcal{N} be a Bayesian network with nodes $1, \dots, n$ and let $\mathcal{D}_{\mathcal{N}}$ be the corresponding class of distributions. The class of concepts induced by \mathcal{N} , denoted as $C_{\mathcal{N}}$, consists of all ± 1 -valued functions on $\{0, 1\}^n$ of the form $\text{sign}(\log(P(x)/Q(x)))$ for $P, Q \in \mathcal{D}_{\mathcal{N}}$.*

Note that the function $\text{sign}(\log(P(x)/Q(x)))$ attains the value 1 if $P(x) \geq Q(x)$, and the value -1 otherwise. We use $\text{VCdim}(\mathcal{N})$ to denote the VC dimension of $C_{\mathcal{N}}$.

2.3 Linear Arrangements in Inner Product Spaces

We are interested in embedding concept classes into finite-dimensional Euclidean spaces equipped with the standard dot product $u^\top v = \sum_{i=1}^d u_i v_i$, where u^\top denotes the transpose of u . Such an embedding is provided by a linear arrangement. Given a concept class C , we aim at determining the smallest Euclidean dimension, denoted $\text{Edim}(C)$, that such a space can have.

Definition 6 *A d -dimensional linear arrangement for a concept class C over domain X is given by collections $(u_f)_{f \in C}$ and $(v_x)_{x \in X}$ of vectors in \mathbb{R}^d such that*

$$\forall f \in C, x \in X : f(x) = \text{sign}(u_f^\top v_x).$$

The smallest d such that there exists a d -dimensional linear arrangement for C is denoted as $\text{Edim}(C)$. If there is no finite-dimensional linear arrangement for C , $\text{Edim}(C)$ is defined to be infinite.

If $C_{\mathcal{N}}$ is the concept class induced by a Bayesian network \mathcal{N} , we write $\text{Edim}(\mathcal{N})$ instead of $\text{Edim}(C_{\mathcal{N}})$. It is evident that $\text{Edim}(C) \leq \text{Edim}(C')$ if $C \leq C'$.

It is easy to see that $\text{Edim}(C) \leq \min\{|C|, |X|\}$ for finite concept classes. Nontrivial upper bounds on $\text{Edim}(C)$ are usually obtained constructively by presenting an appropriate arrangement. As for lower bounds, the following result is immediate from a result by Dudley (1978) which states that $\text{VCdim}(\{\text{sign} \circ f \mid f \in \mathcal{F}\}) = d$ for every d -dimensional vector space \mathcal{F} consisting of real-valued functions (see also Anthony and Bartlett, 1999, Theorem 3.5).

Lemma 7 Every concept class \mathcal{C} satisfies $\text{Edim}(\mathcal{C}) \geq \text{VCdim}(\mathcal{C})$.

Let PARITY_n be the concept class $\{h_a \mid a \in \{0, 1\}^n\}$ of parity functions on the Boolean domain given by $h_a(x) = (-1)^{a^\top x}$, that is, $h_a(x)$ is the parity of those x_i where $a_i = 1$. The following lower bound, which will be useful in Section 5, is due to Forster (2002).

Corollary 8 $\text{Edim}(\text{PARITY}_n) \geq 2^{n/2}$.

3. Upper Bounds on the Dimension of Inner Product Spaces for Bayesian Networks

This section is concerned with the derivation of upper bounds on $\text{Edim}(\mathcal{N})$. We obtain bounds for unconstrained networks and for networks with a reduced parameter collection by providing concrete linear arrangements. Given a set M , let 2^M denote its power set.

Theorem 9 Every unconstrained Bayesian network \mathcal{N} satisfies

$$\text{Edim}(\mathcal{N}) \leq \left| \bigcup_{i=1}^n 2^{P_i \cup \{i\}} \right| \leq 2 \cdot \sum_{i=1}^n 2^{m_i}.$$

Proof From the expansion of P in equation (1) and the corresponding expansion of Q (with parameters $q_{i,\alpha}$ in the role of $p_{i,\alpha}$), we obtain

$$\log \frac{P(x)}{Q(x)} = \sum_{i=1}^n \sum_{\alpha \in \{0,1\}^{m_i}} \left(x_i M_{i,\alpha}(x) \log \frac{p_{i,\alpha}}{q_{i,\alpha}} + (1-x_i) M_{i,\alpha}(x) \log \frac{1-p_{i,\alpha}}{1-q_{i,\alpha}} \right). \quad (3)$$

On the right-hand side of equation (3), we find the polynomials $M_{i,\alpha}(x)$ and $x_i M_{i,\alpha}(x)$. Note that $|\bigcup_{i=1}^n 2^{P_i \cup \{i\}}|$ equals the number of monomials that occur when we express these polynomials as sums of monomials by successive applications of the distributive law. A linear arrangement of the claimed dimensionality is now obtained in the obvious fashion by introducing one coordinate per monomial. ■

This result immediately yields an upper bound for Markov chains of order k .

Corollary 10 Let \mathcal{N}_k be the k th-order Markov chain given in Example 1. Then,

$$\text{Edim}(\mathcal{N}_k) \leq (n-k+1)2^k.$$

Proof Apply Theorem 9 and observe that

$$\bigcup_{i=1}^n 2^{P_i \cup \{i\}} = \bigcup_{i=k+1}^n \{J_i \cup \{i\} \mid J_i \subseteq \{i-1, \dots, i-k\}\} \cup \{J \mid J \subseteq \{1, \dots, k\}\}.$$

Similar techniques as used in the proof of Theorem 9 lead to an upper bound for networks with a reduced parameter collection. ■

Theorem 11 *Let \mathcal{N}^R denote the Bayesian network that has a reduced parameter collection*

$$(P_{i,c})_{1 \leq i \leq n, 1 \leq c \leq d_i}$$

in the sense of Definition 3. Then,

$$\text{Edim}(\mathcal{N}^R) \leq 2 \cdot \sum_{i=1}^n d_i.$$

Proof Recall that the distributions from $\mathcal{D}_{\mathcal{N}^R}$ can be written as in equation (2). We make use of the obvious relationship

$$\log \frac{P(x)}{Q(x)} = \sum_{i=1}^n \sum_{c=1}^{d_i} \left(x_i R_{i,c}(x) \log \frac{P_{i,c}}{q_{i,c}} + (1 - x_i) R_{i,c}(x) \log \frac{1 - P_{i,c}}{1 - q_{i,c}} \right). \quad (4)$$

A linear arrangement of the appropriate dimension is now obtained by introducing two coordinates per pair (i, c) : If x is mapped to v_x in this arrangement, then the projection of v_x to the two coordinates corresponding to (i, c) is $(R_{i,c}(x), x_i R_{i,c}(x))$; the appropriate mapping $(P, Q) \mapsto u_{P,Q}$ in this arrangement is easily derived from (4). ■

In Section 4 we shall show that the bounds established by Theorem 9 and Theorem 11 are tight up to a factor of 2 and, in some cases, even up to an additive constant of 1.

The linear arrangements for unconstrained Bayesian networks or for Bayesian networks with a reduced parameter collection were easy to find. This is no accident as this holds for every class of distributions (or densities) from the so-called exponential family because (as pointed out, for instance, in Devroye et al., 1996) the corresponding Bayes rule takes a form known as generalized linear rule. From this representation a linear arrangement is evident. Note, however, that the bound given in Theorem 9 is slightly stronger than the bound obtained from the general approach for members of the exponential family.

4. Lower Bounds Based on VC Dimension Considerations

In this section, we derive lower bounds on $\text{Edim}(\mathcal{N})$ that come close to the upper bounds obtained in the previous section. Before presenting the main results in Section 4.2 as Corollaries 18, 21, and Theorem 22, we focus on some specific Bayesian networks for which we determine the exact values of $\text{Edim}(\mathcal{N})$.

4.1 Optimal Bounds for Specific Networks

In the following we calculate exact values of $\text{Edim}(\mathcal{N})$ by establishing lower bounds of $\text{VCdim}(\mathcal{N})$ and applying Lemma 7. This gives us also the exact value of the VC dimension for the respective networks. We recall that \mathcal{N}_k is the k th-order Markov chain defined in Example 1. The concept class arising from network \mathcal{N}_0 , which we consider first, is the well-known Naïve Bayes classifier.

Theorem 12

$$\text{Edim}(\mathcal{N}_0) = \begin{cases} n + 1 & \text{if } n \geq 2, \\ 1 & \text{if } n = 1. \end{cases}$$

The proof of this theorem relies on the following result.

Lemma 13 For every $p, q \in]0, 1[$ there exist $w \in \mathbb{R}$ and $b \in]0, 1[$ such that

$$\forall x \in \mathbb{R} : x \log \frac{p}{q} + (1-x) \log \frac{1-p}{1-q} = w(x-b) \tag{5}$$

holds. Conversely, for every $w \in \mathbb{R}$ and $b \in]0, 1[$ there exist $p, q \in]0, 1[$ such that (5) is satisfied.

Proof Rewriting the left-hand side of the equation as $x \log w' + \log c'$, where

$$w' = \frac{p(1-q)}{q(1-p)} \quad \text{and} \quad c' = \frac{1-p}{1-q},$$

it follows that $p = q$ is equivalent to $w' = c' = 1$. By definition of c' , $p < q$ is equivalent to $c' > 1$ and, as $w'c' = p/q$, this is also equivalent to $c' < 1/w'$. Analogously, it follows that $p > q$ is equivalent to $0 < 1/w' < c' < 1$. By defining $w = \log w'$ and $c = \log c'$ and taking logarithms in the equalities and inequalities, we conclude that $p, q \in]0, 1[$ is equivalent to $w \in \mathbb{R}$ and $c = -bw$ with $b \in]0, 1[$. ■

Proof (Theorem 12) Clearly, the theorem holds for $n = 1$. Suppose, therefore, that $n \geq 2$. According to Corollary 10, $\text{Edim}(\mathcal{N}_0) \leq n + 1$. Thus, by Lemma 7 it suffices to show that $\text{VCdim}(\mathcal{N}_0) \geq n + 1$. Let e_i denote the vector with a one in the i th position and zeros elsewhere. Further, let $\bar{1}$ be the vector with a 1 in each position. We show that the set of $n + 1$ vectors $e_1, \dots, e_n, \bar{1}$ is shattered by the class $\mathcal{C}_{\mathcal{N}_0}$ of concepts induced by \mathcal{N}_0 , consisting of the functions of the form

$$\text{sign} \left(\log \frac{P(x)}{Q(x)} \right) = \text{sign} \left(\sum_{i=1}^n x_i \log \frac{p_i}{q_i} + (1-x_i) \log \frac{1-p_i}{1-q_i} \right),$$

where $p_i, q_i \in]0, 1[$, for $i \in \{1, \dots, n\}$. By Lemma 13, the functions in $\mathcal{C}_{\mathcal{N}_0}$ can be written as

$$\text{sign}(w^\top(x-b)),$$

where $w \in \mathbb{R}^n$ and $b \in]0, 1[^n$.

It is not difficult to see that homogeneous halfspaces, that is, where $b = (0, \dots, 0)$, can dichotomize the set $\{e_1, \dots, e_n, \bar{1}\}$ in all possible ways, except for the two cases to separate $\bar{1}$ from e_1, \dots, e_n . To accomplish these two dichotomies we define $b = (3/4) \cdot \bar{1}$ and $w = \pm \bar{1}$. Then, by the assumption that $n \geq 2$, we have for $i = 1, \dots, n$,

$$w^\top(e_i - b) = \pm(1 - 3n/4) \leq 0 \quad \text{and} \quad w^\top(\bar{1} - b) = \pm(n - 3n/4) \geq 0.$$

■

A further type of Bayesian network for which we derive the exact dimension has some kind of bipartite graph underlying where one set of nodes serves as the set of parents for all nodes in the other set.

Theorem 14 For $k \geq 0$, let \mathcal{N}_k' denote the unconstrained network with $P_i = \emptyset$ for $i = 1, \dots, k$ and $P_i = \{1, \dots, k\}$ for $i = k + 1, \dots, n$. Then,

$$\text{Edim}(\mathcal{N}_k') = 2^k(n - k + 1).$$

Proof For the upper bound, we apply Lemma 1 and Theorem 1 using the fact that

$$\bigcup_{i=1}^n 2^{P_i \cup \{x_i\}} = \bigcup_{i=k+1}^n \{J_i \cup \{i\} \mid J_i \subseteq \{1, \dots, k\}\} \cup \{J \mid J \subseteq \{1, \dots, k\}\}.$$

To obtain the lower bound, let $M \subseteq \{0, 1\}^{n-k}$ denote the set from the proof of Theorem 12 for the corresponding network \mathcal{N}_0 with $n - k$ nodes. We show that the set $S = \{0, 1\}^k \times M \subseteq \{0, 1\}^n$ is shattered by \mathcal{N}_k' . Note that S has the claimed cardinality since $|M| = n - k + 1$.

Let (S^-, S^+) be a dichotomy of S (that is, where $S^- \cup S^+ = S$ and $S^- \cap S^+ = \emptyset$). Given a natural number $j \in \{0, \dots, 2^k - 1\}$, we use $\text{bin}(j)$ to denote the binary representation of j using k bits. Then, let (M_j^-, M_j^+) be the dichotomy of M defined by

$$M_j^+ = \{v \in M \mid \text{bin}(j)v \in S^+\}.$$

Here, $\text{bin}(j)v$ refers to the concatenation of the k bits of $\text{bin}(j)$ and the $n - k$ bits of v . According to Theorem 12, for each dichotomy (M_j^-, M_j^+) there exist parameter values p_i^j, q_i^j , where $1 \leq i \leq n - k$, such that \mathcal{N}_0 with these parameter settings induces this dichotomy on M . In the network \mathcal{N}_k' , we specify the parameters as follows. For $i = 1, \dots, k$, let

$$p_i = q_i = 1/2,$$

and for $i = k + 1, \dots, n$ and each $j \in \{0, \dots, 2^k - 1\}$ define

$$\begin{aligned} p_{i, \text{bin}(j)} &= p_{i-k}^j, \\ q_{i, \text{bin}(j)} &= q_{i-k}^j. \end{aligned}$$

Obviously, the concept thus defined by \mathcal{N}_k' outputs -1 for elements of S^- and 1 for elements of S^+ . Since every dichotomy of S can be implemented in this way, S is shattered by \mathcal{N}_k' . ■

4.2 General Lower Bounds

In Section 4.2.1 we shall establish lower bounds on $\text{Edim}(\mathcal{N})$ for unconstrained Bayesian networks and in Section 4.2.2 for networks with a reduced parameter collection. These results are obtained by providing embeddings of concept classes, as introduced in Section 2.1, into these networks. Since $\text{VCdim}(\mathcal{C}) \leq \text{VCdim}(\mathcal{C}')$ if $\mathcal{C} \leq \mathcal{C}'$, a lower bound on $\text{VCdim}(\mathcal{C}')$ follows immediately from classes satisfying $\mathcal{C} \leq \mathcal{C}'$ if the VC dimension of \mathcal{C} is known or easy to determine. We first define concept classes that will suit this purpose.

Definition 15 Let \mathcal{N} be an arbitrary Bayesian network. For every $i \in \{1, \dots, n\}$, let \mathcal{F}_i be a family of ± 1 -valued functions on the domain $\{0, 1\}^{m_i}$ and let $\mathcal{F} = \mathcal{F}_1 \times \dots \times \mathcal{F}_n$. Then $\mathcal{C}_{\mathcal{N}, \mathcal{F}}$ is the concept class over the domain $\{0, 1\}^n \setminus \{(0, \dots, 0)\}$ consisting of all functions of the form

$$L_{\mathcal{N}, \mathcal{F}} = [(x_n, f_n), \dots, (x_1, f_1)],$$

where $f = (f_1, \dots, f_n) \in \mathcal{F}$. The right-hand side of this equation is to be understood as a decision list, where $L_{\mathcal{N},f}(x)$ for $x \neq (0, \dots, 0)$ is determined as follows:

1. Find the largest i such that $x_i = 1$.
2. Apply f_i to the projection of x to the parent-variables of x_i and output the result.

The VC dimension of $C_{\mathcal{N},\mathcal{F}}$ can be directly obtained from the VC dimensions of the classes \mathcal{F}_i .

Lemma 16 *Let \mathcal{N} be an arbitrary Bayesian network. Then,*

$$\text{VCdim}(C_{\mathcal{N},\mathcal{F}}) = \sum_{i=1}^n \text{VCdim}(\mathcal{F}_i).$$

Proof We show that $\text{VCdim}(C_{\mathcal{N},\mathcal{F}}) \geq \sum_{i=1}^n \text{VCdim}(\mathcal{F}_i)$; the proof for the other direction is similar. For every i , we embed the vectors from $\{0, 1\}^{m_i}$ into $\{0, 1\}^n$ according to $\tau_i(a) = (a', 1, 0, \dots, 0)$, where $a' \in \{0, 1\}^{i-1}$ is chosen such that its projection to the parent-variables of x_i is equal to a and the remaining components are set to 0. Note that $\tau_i(a)$ is absorbed by item (x_i, f_i) of the decision list $L_{\mathcal{N},f}$. It is easy to see that the following holds: If, for $i = 1, \dots, n$, S_i is a set that is shattered by \mathcal{F}_i , then $\cup_{i=1}^n \tau_i(S_i)$ is shattered by $C_{\mathcal{N},\mathcal{F}}$. Thus, $\text{VCdim}(C_{\mathcal{N},\mathcal{F}}) \geq \sum_{i=1}^n \text{VCdim}(\mathcal{F}_i)$. ■

The preceding definition and lemma are valid for unconstrained as well as constrained networks as they make use only of the graph underlying the network and do not refer to the values of the parameters. This will be important in the applications that follow.

4.2.1 LOWER BOUNDS FOR UNCONSTRAINED BAYESIAN NETWORKS

The next theorem is the main step in deriving for an arbitrary unconstrained network \mathcal{N} a lower bound on $\text{Edim}(\mathcal{N})$. It is based on the idea of embedding one of the concept classes $C_{\mathcal{N},\mathcal{F}}$ defined above into $C_{\mathcal{N}}$.

Theorem 17 *Let \mathcal{N} be an unconstrained Bayesian network and let \mathcal{F}_i^* denote the set of all ± 1 -valued functions on domain $\{0, 1\}^{m_i}$. Further, let $\mathcal{F}^* = \mathcal{F}_1^* \times \dots \times \mathcal{F}_n^*$. Then, $C_{\mathcal{N},\mathcal{F}^*} \leq C_{\mathcal{N}}$.*

Proof We have to show that, for every $f = (f_1, \dots, f_n)$, we can find a pair (P, Q) of distributions from $\mathcal{D}_{\mathcal{N}}$ such that, for every $x \in \{0, 1\}^n$, $L_{\mathcal{N},f}(x) = \text{sign}(\log(P(x)/Q(x)))$. To this end, we define the parameters for the distributions P and Q as

$$p_{i,\alpha} = \begin{cases} 2^{-2^{i-1}n}/2 & \text{if } f_i(\alpha) = -1, \\ 1/2 & \text{if } f_i(\alpha) = +1, \end{cases} \quad \text{and} \quad q_{i,\alpha} = \begin{cases} 1/2 & \text{if } f_i(\alpha) = -1, \\ 2^{-2^{i-1}n}/2 & \text{if } f_i(\alpha) = +1. \end{cases}$$

An easy calculation now shows that

$$\log\left(\frac{p_{i,\alpha}}{q_{i,\alpha}}\right) = f_i(\alpha)2^{i-1}n \quad \text{and} \quad \left|\log\frac{1-p_{i,\alpha}}{1-q_{i,\alpha}}\right| < 1. \quad (6)$$

Fix some arbitrary $x \in \{0, 1\}^n \setminus \{(0, \dots, 0)\}$. Choose i_* maximal such that $x_{i_*} = 1$ and let α_* denote the projection of x to the parent-variables of x_{i_*} . Then, $L_{\mathcal{N},f}(x) = f_{i_*}(\alpha_*)$. Thus, $L_{\mathcal{N},f}(x) = \text{sign}(\log(P(x)/Q(x)))$ would follow immediately from

$$\text{sign}\left(\log\frac{P(x)}{Q(x)}\right) = \text{sign}\left(\log\frac{p_{i_*,\alpha_*}}{q_{i_*,\alpha_*}}\right) = f_{i_*}(\alpha_*). \quad (7)$$

The second equation in (7) is evident from the equality established in (6). As for the first equation in (7), we argue as follows. By the choice of i_* , we have $x_i = 0$ for every $i > i_*$. Expanding P and Q as given in (3), we obtain

$$\begin{aligned} \log \frac{P(x)}{Q(x)} &= \log \frac{p_{i_*, \alpha_*}}{q_{i_*, \alpha_*}} + \sum_{i=1}^{i_*-1} \left(\sum_{\alpha \in \{0,1\}^{m_i}} x_i M_{i,\alpha}(x) \log \frac{p_{i,\alpha}}{q_{i,\alpha}} \right) \\ &\quad + \sum_{i \in I} \left(\sum_{\alpha \in \{0,1\}^{m_i}} (1-x_i) M_{i,\alpha}(x) \log \frac{1-p_{i,\alpha}}{1-q_{i,\alpha}} \right), \end{aligned}$$

where $I = \{1, \dots, n\} \setminus \{i_*\}$. Employing the inequality from (6), it follows that the sign of the right-hand side of this equation is determined by $\log(p_{i_*, \alpha_*}/q_{i_*, \alpha_*})$ since this term is of absolute value $2^{i_*-1}n$ and

$$2^{i_*-1}n - \sum_{j=1}^{i_*-1} (2^{j-1}n) - (n-1) \geq 1. \tag{8}$$

This concludes the proof. ■

Using the lower bound obtained from Theorem 17 combined with Lemma 16 and the upper bound provided by Theorem 9, we have a result that is tight up to a factor of 2.

Corollary 18 *Every unconstrained Bayesian network \mathcal{N} satisfies*

$$\sum_{i=1}^n 2^{m_i} \leq \text{Edim}(\mathcal{N}) \leq \left| \bigcup_{i=1}^n 2^{P_i \cup \{i\}} \right| \leq 2 \cdot \sum_{i=1}^n 2^{m_i}.$$

Bounds for the k th-order Markov chain that are optimal up to an additive constant of 1 emerge from the lower bound due to Theorem 17 with Lemma 16 and the upper bound stated in Corollary 10.

Corollary 19 *Let \mathcal{N}_k denote the Bayesian network from Example 1. Then,*

$$(n-k+1)2^k - 1 \leq \text{Edim}(\mathcal{N}_k) \leq (n-k+1)2^k.$$

4.2.2 LOWER BOUNDS FOR BAYESIAN NETWORKS WITH A REDUCED PARAMETER COLLECTION

We now show how to obtain bounds for networks with a reduced parameter collection. Similarly as in Section 4.2.1, the major step consists in providing embeddings into these networks. The main result is based on techniques developed for Theorem 17.

Theorem 20 *Let \mathcal{N}^R denote the Bayesian network that has a reduced parameter collection*

$$(p_{i,c})_{1 \leq i \leq n, 1 \leq c \leq d_i}$$

in the sense of Definition 3. Let $\mathcal{F}_i^{R_i}$ denote the set of all ± 1 -valued functions on the domain $\{0, 1\}^{m_i}$ that depend on $\alpha \in \{0, 1\}^{m_i}$ only through $R_i(\alpha)$. In other words, $f \in \mathcal{F}_i^{R_i}$ holds if and only if there exists a ± 1 -valued function g on domain $\{1, \dots, d_i\}$ such that $f(\alpha) = g(R_i(\alpha))$ for every $\alpha \in \{0, 1\}^{m_i}$. Finally, let $\mathcal{F}^R = \mathcal{F}_1^{R_1} \times \dots \times \mathcal{F}_n^{R_n}$. Then, $C_{\mathcal{N}^R, \mathcal{F}^R} \leq C_{\mathcal{N}^R}$.

Proof We focus on the differences to the proof of Theorem 17. First, the decision list $L_{\mathcal{N}^R, f}$ uses a function $f = (f_1, \dots, f_n)$ of the form $f_i(x) = g_i(R_i(x))$ for some function $g_i : \{1, \dots, d_i\} \rightarrow \{-1, 1\}$. Second, the distributions P, Q that satisfy $L_{\mathcal{N}, f}(x) = \text{sign}(\log(P(x)/Q(x)))$ for every $x \in \{0, 1\}^n$ have to be defined over the reduced parameter collection as given in equation (4). An appropriate choice is

$$p_{i,c} = \begin{cases} 2^{-2^{i-1}n}/2 & \text{if } g_i(c) = -1, \\ 1/2 & \text{if } g_i(c) = 1, \end{cases} \quad \text{and} \quad q_{i,c} = \begin{cases} 1/2 & \text{if } g_i(c) = -1, \\ 2^{-2^{i-1}n}/2 & \text{if } g_i(c) = 1. \end{cases}$$

The rest of the proof is completely analogous to the proof of Theorem 17. ■

Theorem 17 can be viewed as a special case of Theorem 20 since every unconstrained network can be considered as a network with a reduced parameter collection where the functions R_i are 1-1. However, there are differences arising from the notation of the network parameters that have been taken into account by the above proof.

Applying the lower bound of Theorem 20 in combination with Lemma 16 and the upper bound of Theorem 11, we once more have bounds that are optimal up to the factor 2.

Corollary 21 *Let \mathcal{N}^R denote the Bayesian network that has a reduced parameter collection*

$$(p_{i,c})_{1 \leq i \leq n, 1 \leq c \leq d_i}$$

in the sense of Definition 3. Then,

$$\sum_{i=1}^n d_i \leq \text{Edim}(\mathcal{N}^R) \leq 2 \cdot \sum_{i=1}^n d_i.$$

4.2.3 LOWER BOUNDS FOR LOGISTIC AUTOREGRESSIVE NETWORKS

The following result is not obtained by embedding a concept class into a logistic autoregressive Bayesian network. However, we apply a similar technique as developed in Sections 4.2.1 and 4.2.2 to derive a bound using the VC dimension by directly showing that these networks can shatter sets of the claimed size.

Theorem 22 *Let $\mathcal{N}_{\mathcal{G}}$ denote the logistic autoregressive Bayesian network from Definition 4. Then,*

$$\text{Edim}(\mathcal{N}_{\mathcal{G}}) \geq n(n-1)/2.$$

Proof We show that the following set S is shattered by the concept class $\mathcal{C}_{\mathcal{N}_{\mathcal{G}}}$. Then the statement follows from Lemma 7.

For $i = 2, \dots, n$ and $c = 1, \dots, i-1$, let $\alpha_{i,c} \in \{0, 1\}^{i-1}$ be the pattern with bit 1 in position c and zeros elsewhere. Then, for every pair (i, c) , where $i \in \{2, \dots, n\}$ and $c \in \{1, \dots, i-1\}$, let $s^{(i,c)} \in \{0, 1\}^n$ be the vector that has bit 1 in coordinate i , bit-pattern $\alpha_{i,c}$ in the coordinates $1, \dots, i-1$, and zeros in the remaining positions. The set

$$S = \{s^{(i,c)} \mid i = 2, \dots, n \text{ and } c = 1, \dots, i-1\}$$

has $n(n-1)/2$ elements.

To show that S is shattered, let (S^-, S^+) be some arbitrary dichotomy of S . We claim that there exists a pair (P, Q) of distributions from $\mathcal{D}_{\mathcal{N}_G}$ such that for every $s^{(i,c)}$, $\text{sign}(\log(P(s^{(i,c)})/Q(s^{(i,c)}))) = 1$ if and only if $s^{(i,c)} \in S^+$. Assume that the parameters $p_{i,\alpha}$ and $q_{i,\alpha}$ for the distributions P and Q , respectively, satisfy

$$p_{i,\alpha} = \begin{cases} 1/2 & \text{if } \alpha = \alpha_{i,c} \text{ and } s^{(i,c)} \in S^+, \\ 2^{-2^{i-1}n}/2 & \text{otherwise,} \end{cases}$$

and

$$q_{i,\alpha} = \begin{cases} 2^{-2^{i-1}n}/2 & \text{if } \alpha = \alpha_{i,c} \text{ and } s^{(i,c)} \in S^+, \\ 1/2 & \text{otherwise.} \end{cases}$$

Similarly as in the proof of Theorem 17, we have

$$\left| \log \left(\frac{p_{i,\alpha}}{q_{i,\alpha}} \right) \right| = 2^{i-1}n \quad \text{and} \quad \left| \log \frac{1-p_{i,\alpha}}{1-q_{i,\alpha}} \right| < 1. \tag{9}$$

The expansion of P and Q yields for every $s^{(i,c)} \in S$,

$$\begin{aligned} \log \frac{P(s^{(i,c)})}{Q(s^{(i,c)})} &= \log \frac{p_{i,\alpha_{i,c}}}{q_{i,\alpha_{i,c}}} + \sum_{j=1}^{i-1} \left(\sum_{\alpha \in \{0,1\}^{j-1}} s_j^{(i,c)} M_{j,\alpha}(s^{(i,c)}) \log \frac{p_{j,\alpha}}{q_{j,\alpha}} \right) \\ &\quad + \sum_{j \in I} \left(\sum_{\alpha \in \{0,1\}^{j-1}} (1-s_j^{(i,c)}) M_{j,\alpha}(s^{(i,c)}) \log \frac{1-p_{j,\alpha}}{1-q_{j,\alpha}} \right), \end{aligned}$$

where $I = \{1, \dots, n\} \setminus \{i\}$. In analogy to inequality (8) in the proof of Theorem 17, it follows from (9) that the sign of $\log(P(s^{(i,c)})/Q(s^{(i,c)}))$ is equal to the sign of $\log(p_{i,\alpha_{i,c}}/q_{i,\alpha_{i,c}})$. By the definition of $p_{i,\alpha_{i,c}}$ and $q_{i,\alpha_{i,c}}$, the sign of $\log(p_{i,\alpha_{i,c}}/q_{i,\alpha_{i,c}})$ is positive if and only if $s^{(i,c)} \in S^+$.

It remains to show that the parameters of the distributions P and Q can be given as required by Definition 4, that is, in the form $p_{i,\alpha} = \sigma(\sum_{j=1}^{i-1} w_{i,j} \alpha_j)$ with $w_{i,j} \in \mathbb{R}$, and similarly for $q_{i,\alpha}$. This now immediately follows from the fact that $\sigma(\mathbb{R}) =]0, 1[$. ■

5. Lower Bounds via Embeddings of Parity Functions

The lower bounds obtained in Section 4 rely on arguments based on the VC dimension of the respective concept class. In particular, a quadratic lower bound for the logistic autoregressive network has been established. In the following, we introduce a different technique leading to the lower bound $2^{\Omega(n)}$ for a variant of this network. For the time being, it seems possible to obtain an exponential bound for these slightly modified networks only, which are given by the following definition.

Definition 23 *The modified logistic autoregressive Bayesian network \mathcal{N}_G^l is the fully connected Bayesian network with nodes $0, 1, \dots, n+1$ and the constraints on the parameter collection defined as*

$$\forall i = 0, \dots, n, \exists (w_{i,j})_{0 \leq j \leq i-1} \in \mathbb{R}^i, \forall \alpha \in \{0, 1\}^i : p_{i,\alpha} = \sigma \left(\sum_{j=0}^{i-1} w_{i,j} \alpha_j \right)$$

and

$$\exists (w_i)_{0 \leq i \leq n}, \forall \alpha \in \{0, 1\}^{n+1} : p_{n+1, \alpha} = \sigma \left(\sum_{i=0}^n w_i \sigma \left(\sum_{j=0}^{i-1} w_{i,j} \alpha_j \right) \right).$$

Obviously, $\mathcal{N}'_{\mathcal{G}}$ is completely described by the parameter collections $(w_{i,j})_{0 \leq i \leq n, 0 \leq j \leq i-1}$ and $(w_i)_{0 \leq i \leq n}$.

The crucial difference between $\mathcal{N}'_{\mathcal{G}}$ and $\mathcal{N}_{\mathcal{G}}$ is the node $n + 1$ whose sigmoidal function receives the outputs of the other sigmoidal functions as input. Roughly speaking, $\mathcal{N}_{\mathcal{G}}$ is a single-layer network whereas $\mathcal{N}'_{\mathcal{G}}$ has an extra node at a second layer.

To obtain the bound, we provide an embedding of the concept class of parity functions. The following theorem motivates this construction by showing that it is impossible to obtain an exponential lower bound for $\text{Edim}(\mathcal{N}_{\mathcal{G}})$ nor for $\text{Edim}(\mathcal{N}'_{\mathcal{G}})$ using the VC dimension argument, as these networks have VC dimensions that are polynomial in n .

Theorem 24 *The logistic autoregressive Bayesian network $\mathcal{N}_{\mathcal{G}}$ from Definition 4 and the modified logistic autoregressive Bayesian network $\mathcal{N}'_{\mathcal{G}}$ from Definition 23 have a VC dimension that is bounded by $O(n^6)$.*

Proof Consider first the logistic autoregressive Bayesian network. We show that the concept class induced by $\mathcal{N}_{\mathcal{G}}$ can be computed by a specific type of feedforward neural network. Then, we apply a known bound on the VC dimension of these networks.

The neural networks for the concepts in $\mathcal{C}_{\mathcal{N}_{\mathcal{G}}}$ consist of sigmoidal units, product units, and units computing second-order polynomials. A sigmoidal unit computes functions of the form $\sigma(w^\top x - t)$, where $x \in \mathbb{R}^k$ is the input vector and $w \in \mathbb{R}^k, t \in \mathbb{R}$ are parameters. A product unit computes $\prod_{i=1}^k x_i^{w_i}$.

The value of $p_{i,\alpha}$ can be calculated by a sigmoidal unit as $p_{i,\alpha} = \sigma(\sum_{j=1}^{i-1} w_{i,j} \alpha_j)$ with α as input and parameters $w_{i,1}, \dots, w_{i,i-1}$. Regarding the factors $p_{i,\alpha}^{x_i} (1 - p_{i,\alpha})^{(1-x_i)}$, we observe that

$$\begin{aligned} p_{i,\alpha}^{x_i} (1 - p_{i,\alpha})^{(1-x_i)} &= p_{i,\alpha} x_i + (1 - p_{i,\alpha})(1 - x_i) \\ &= 2p_{i,\alpha} x_i - x_i - p_{i,\alpha} + 1, \end{aligned}$$

where the first equation is valid because $x_i \in \{0, 1\}$. Thus, the value of $p_{i,\alpha}^{x_i} (1 - p_{i,\alpha})^{(1-x_i)}$ is given by a second-order polynomial. Similarly, the value of $q_{i,\alpha}^{x_i} (1 - q_{i,\alpha})^{(1-x_i)}$ can also be determined using sigmoidal units and polynomial units of order 2. Finally, the output value of the network is obtained by comparing $P(x)/Q(x)$ with the constant threshold 1. We calculate $P(x)/Q(x)$ using a product unit

$$y_1 \cdots y_n z_1^{-1} \cdots z_n^{-1},$$

with input variables y_i and z_i that receive the value of $p_{i,\alpha}^{x_i} (1 - p_{i,\alpha})^{(1-x_i)}$ and $q_{i,\alpha}^{x_i} (1 - q_{i,\alpha})^{(1-x_i)}$ computed by the second-order units, respectively.

This network has $O(n^2)$ parameters and $O(n)$ computation nodes, each of which is a sigmoidal unit, a second-order unit, or a product unit. Theorem 2 of Schmitt (2002) shows that every such network with W parameters and k computation nodes, which are sigmoidal and product units, has VC dimension $O(W^2 k^2)$. A close inspection of the proof of this result reveals that it also includes polynomials of degree 2 as computational units (see also Lemma 4 in Schmitt, 2002). Thus, we obtain the claimed bound $O(n^6)$ for the logistic autoregressive Bayesian network $\mathcal{N}_{\mathcal{G}}$.

For the modified logistic autoregressive network we have only to take one additional sigmoidal unit into account. Thus, the bound for this network follows now immediately. ■

In the previous result we were interested in the asymptotic behavior of the VC dimension, showing that it is not exponential. Using the techniques provided in Schmitt (2002) mentioned in the above proof, it is also possible to obtain constant factors for these bounds.

We now provide the main result of this section. Its proof employs the concept class PARITY_n defined in Section 2.3.

Theorem 25 *Let $\mathcal{N}'_{\mathcal{G}}$ denote the modified logistic autoregressive Bayesian network with $n + 2$ nodes and assume that n is a multiple of 4. Then, $\text{PARITY}_{n/2} \leq \mathcal{N}'_{\mathcal{G}}$.*

Proof The mapping

$$\{0, 1\}^{n/2} \ni x = (x_1, \dots, x_{n/2}) \mapsto (\overbrace{1, x_1, \dots, x_{n/2}, 1, \dots, 1}^{\alpha}, 1) = x' \in \{0, 1\}^{n+2} \quad (10)$$

assigns to every element of $\{0, 1\}^{n/2}$ uniquely some element in $\{0, 1\}^{n+2}$. Note that α , as indicated in (10), equals the bit-pattern of the parent-variables of x'_{n+2} (which are actually all other variables). We claim that the following holds. For every $a \in \{0, 1\}^{n/2}$, there exists a pair (P, Q) of distributions from $\mathcal{D}_{\mathcal{N}'_{\mathcal{G}}}$ such that for every $x \in \{0, 1\}^{n/2}$,

$$(-1)^{a^\top x} = \text{sign} \left(\log \frac{P(x')}{Q(x')} \right). \quad (11)$$

Clearly, the theorem follows once the claim is settled. The proof of the claim makes use of the following facts:

Fact 1 For every $a \in \{0, 1\}^{n/2}$, function $(-1)^{a^\top x}$ can be computed by a two-layer threshold circuit with $n/2$ threshold units at the first layer and one threshold unit as output node at the second layer.

Fact 2 Each two-layer threshold circuit C can be simulated by a two-layer sigmoidal circuit C' with the same number of units and the following output convention: $C(x) = 1 \implies C'(x) \geq 2/3$ and $C(x) = 0 \implies C'(x) \leq 1/3$.

Fact 3 Network $\mathcal{N}'_{\mathcal{G}}$ contains as a sub-network a two-layer sigmoidal circuit C' with $n/2$ input nodes, $n/2$ sigmoidal units at the first layer, and one sigmoidal unit at the second layer.

The parity function is a symmetric Boolean function, that is, a function $f : \{0, 1\}^k \rightarrow \{0, 1\}$ that is described by a set $M \subseteq \{0, \dots, k\}$ such that $f(x) = 1$ if and only if $\sum_{i=1}^k x_i \in M$. Thus, Fact 1 is implied by Proposition 2.1 of Hajnal et al. (1993) which shows that every symmetric Boolean function can be computed by a circuit of this kind.

Fact 2 follows from the capability of the sigmoidal function σ to approximate any Boolean threshold function arbitrarily close. This can be done by multiplying all weights and the threshold with a sufficiently large number.

To establish Fact 3, we refer to Definition 23 and proceed as follows: We would like the term $p_{n+1, \alpha}$ to satisfy $p_{n+1, \alpha} = C'(\alpha_1, \dots, \alpha_{n/2})$, where C' denotes an arbitrary two-layer sigmoidal circuit as described in Fact 3. To this end, we set $w_{i,j} = 0$ if $1 \leq i \leq n/2$ or if $i, j \geq n/2 + 1$. Further, we

let $w_i = 0$ if $1 \leq i \leq n/2$. The parameters that have been set to zero are referred to as “redundant” parameters in what follows. Recall from (10) that $\alpha_0 = \alpha_{n/2+1} = \dots = \alpha_n = 1$. From these settings and from $\sigma(0) = 1/2$, we obtain

$$p_{n+1,\alpha} = \sigma \left(\frac{1}{2}w_0 + \sum_{i=n/2+1}^n w_i \sigma \left(w_{i,0} + \sum_{j=1}^{n/2} w_{i,j} \alpha_j \right) \right).$$

Indeed, this is the output of a two-layer sigmoidal circuit C' on the input $(\alpha_1, \dots, \alpha_{n/2})$.

We are now in the position to describe the choice of distributions P and Q . Let C' be the sigmoidal circuit that computes $(-1)^{a^\top x}$ for some fixed $a \in \{0, 1\}^{n/2}$ according to Facts 1 and 2. Let P be the distribution obtained by setting the redundant parameters to zero (as described above) and the remaining parameters as in C' . Thus, $p_{n+1,\alpha} = C'(\alpha_1, \dots, \alpha_{n/2})$. Let Q be the distribution with the same parameters as P except for replacing w_i by $-w_i$. Thus, by symmetry of σ , $q_{n+1,\alpha} = 1 - C'(\alpha_1, \dots, \alpha_{n/2})$. Since $x'_{n+1} = 1$ and since all but one factor in $P(x')/Q(x')$ cancel each other, we arrive at

$$\frac{P(x')}{Q(x')} = \frac{p_{n+1,\alpha}}{q_{n+1,\alpha}} = \frac{C'(\alpha_1, \dots, \alpha_{n/2})}{1 - C'(\alpha_1, \dots, \alpha_{n/2})}.$$

As C' computes $(-1)^{a^\top x}$, the output convention from Fact 2 yields that $P(x')/Q(x') \geq 2$ if $(-1)^{a^\top x} = 1$, and $P(x')/Q(x') \leq 1/2$ otherwise. This implies claim (11) and concludes the proof. \blacksquare

Combining Theorem 25 with Corollary 8, we obtain the exponential lower bound for the modified logistic autoregressive Bayesian network.

Corollary 26 *Let $\mathcal{N}'_{\mathcal{G}}$ denote the modified logistic autoregressive Bayesian network. Then,*

$$\text{Edim}(\mathcal{N}'_{\mathcal{G}}) \geq 2^{n/4}.$$

By a more detailed analysis it can be shown that Theorem 25 holds even if we restrict the values in the parameter collection of $\mathcal{N}'_{\mathcal{G}}$ to integers that can be represented using $O(\log n)$ bits. We mentioned in the introduction that a large lower bound on $\text{Edim}(C)$ rules out the possibility of a large margin classifier. Forster and Simon (2002) have shown that every linear arrangement for PARITY_n has an average geometric margin of at most $2^{-n/2}$. Thus there can be no linear arrangement with an average margin exceeding $2^{-n/4}$ for $C_{\mathcal{N}'_{\mathcal{G}}}$ even if we restrict the weight parameters in $\mathcal{N}'_{\mathcal{G}}$ to logarithmically bounded integers.

6. Conclusions and Open Problems

Bayesian networks have become one of the heavily studied and widely used probabilistic techniques for pattern recognition and statistical inference. One line of inquiry into Bayesian networks pursues the idea of combining them with kernel methods so that one can take advantage of both. Kernel methods employ the principle of mapping the input vectors to some higher-dimensional space where then inner product operations are performed implicitly. The major motivation for our work was to reveal more about such inner product spaces. In particular, we asked whether Bayesian networks can be considered as linear classifiers and, thus, whether kernel operations can be implemented as standard dot products. With this work we have gained insight into the nature of the inner product

space in terms of bounds on its dimensionality. As the main results, we have established tight bounds on the Euclidean dimension of spaces in which two-label classifications of Bayesian networks with binary nodes can be implemented.

We have employed the VC dimension as one of the tools for deriving lower bounds. Bounds on the VC dimension of concept classes abound. Exact values are known only for a few classes. Surprisingly, our investigation of the dimensionality of embeddings lead to some exact values of the VC dimension for nontrivial Bayesian networks. The VC dimension can be employed to obtain tight bounds on the complexity of model selection, that is, on the amount of information required for choosing a Bayesian network that performs well on unseen data. In frameworks where this amount can be expressed in terms of the VC dimension, the tight bounds for the embeddings of Bayesian networks established here show that the sizes of the training samples required for learning can also be estimated using the Euclidean dimension. Another consequence of this close relationship between VC dimension and Euclidean dimension is that these networks can be replaced by linear classifiers without a significant increase in the required sample sizes. Whether these conclusions can be drawn also for the logistic autoregressive network is an open issue. It remains to be shown if the VC dimension is also useful in tightly bounding the Euclidean dimension of these networks. For the modified version of this model, our results suggest that different approaches might be more successful.

The results raise some further open questions. First, since we considered only networks with binary nodes, analogous questions regarding Bayesian networks with multiple-valued nodes or even continuous-valued nodes are certainly of interest. Another generalization of Bayesian networks are those with hidden variables which have also been out of the scope of this work. Further, with regard to logistic autoregressive Bayesian networks, we were able to obtain an exponential lower bound only for a variant of them. For the unmodified network such a bound has yet to be found. Finally, the questions we studied here are certainly relevant not only for Bayesian networks but also for other popular classes of distributions or densities. Those from the exponential family look like a good thing to start with.

Acknowledgments

We thank the anonymous reviewers for the many helpful and comprehensive comments.

This work was supported in part by the IST Programme of the European Community under the PASCAL Network of Excellence, IST-2002-506778, by the Deutsche Forschungsgemeinschaft (DFG), grant SI 498/7-1, and by the “Wilhelm und Günter Esser Stiftung”, Bochum.

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden Markov support vector machines. In *Proceedings of the 20th International Conference on Machine Learning*, pages 3–10. AAAI Press, Menlo Park, CA, 2003.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, 1999.

- Rosa I. Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 616–623. IEEE Computer Society Press, Los Alamitos, CA, 1999.
- Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. On kernels, margins, and low-dimensional mappings. In Shai Ben-David, John Case, and Akira Maruoka, editors, *Proceedings of the 15th International Conference on Algorithmic Learning Theory ALT 2004*, volume 3244 of *Lecture Notes in Artificial Intelligence*, pages 194–205. Springer-Verlag, Berlin, 2004.
- Shai Ben-David, Nadav Eiron, and Hans Ulrich Simon. Limitations of learning via embeddings in Euclidean half-spaces. *Journal of Machine Learning Research*, 3:441–461, 2002.
- Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, New York, NY, 1992.
- David Maxwell Chickering, David Heckerman, and Christopher Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 80–89. Morgan Kaufmann, San Francisco, CA, 1997.
- Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, Berlin, 1996.
- Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, NY, 1973.
- R. M. Dudley. Central limit theorems for empirical measures. *Annals of Probability*, 6:899–929, 1978.
- Jürgen Forster. A linear lower bound on the unbounded error communication complexity. *Journal of Computer and System Sciences*, 65:612–625, 2002.
- Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *Proceedings of the 21st Annual Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 2245 of *Lecture Notes in Computer Science*, pages 171–182. Springer-Verlag, Berlin, 2001.
- Jürgen Forster, Niels Schmitt, Hans Ulrich Simon, and Thorsten Suttrop. Estimating the optimal margins of embeddings in Euclidean halfspaces. *Machine Learning*, 51:263–281, 2003.
- Jürgen Forster and Hans Ulrich Simon. On the smallest possible dimension and the largest possible margin of linear arrangements representing given concept classes. In Nicolò Cesa-Bianchi, Masayuki Numao, and Rüdiger Reischuk, editors, *Proceedings of the 13th International Workshop on Algorithmic Learning Theory ALT 2002*, volume 2533 of *Lecture Notes in Artificial Intelligence*, pages 128–138. Springer-Verlag, Berlin, 2002.
- P. Frankl and H. Maehara. The Johnson-Lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory, Series B*, 44:355–362, 1988.

- Brendan J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge, MA, 1998.
- Andras Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and Györgi Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46:129–154, 1993.
- Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In Michael S. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, Cambridge, MA, 1999a.
- Tommi S. Jaakkola and David Haussler. Probabilistic kernel regression models. In David Heckerman and Joe Whittaker, editors, *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann, San Francisco, CA, 1999b.
- W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mapping into Hilbert spaces. *Contemporary Mathematics*, 26:189–206, 1984.
- Eike Kiltz. On the representation of Boolean predicates of the Diffie-Hellman function. In H. Alt and M. Habib, editors, *Proceedings of 20th International Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 223–233. Springer-Verlag, Berlin, 2003.
- Eike Kiltz and Hans Ulrich Simon. Complexity theoretic aspects of some cryptographic functions. In T. Warnow and B. Zhu, editors, *Proceedings of the 9th International Conference on Computing and Combinatorics COCOON 2003*, volume 2697 of *Lecture Notes in Computer Science*, pages 294–303. Springer-Verlag, Berlin, 2003.
- P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1983.
- Atsuyoshi Nakamura, Michael Schmitt, Niels Schmitt, and Hans Ulrich Simon. Bayesian networks and inner product spaces. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the 17th Annual Conference on Learning Theory COLT 2004*, volume 3120 of *Lecture Notes in Artificial Intelligence*, pages 518–533. Springer-Verlag, Berlin, 2004.
- Radford M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.
- Nuria Oliver, Bernhard Schölkopf, and Alexander J. Smola. Natural regularization from generative models. In Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 51–60. MIT Press, Cambridge, MA, 2000.
- Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the National Conference on Artificial Intelligence*, pages 133–136. AAAI Press, Menlo Park, CA, 1982.
- Laurence K. Saul, Tommi Jaakkola, and Michael I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.
- Craig Saunders, John Shawe-Taylor, and Alexei Vinokourov. String kernels, Fisher kernels and finite state automata. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 633–640. MIT Press, Cambridge, MA, 2003.

- Michael Schmitt. On the complexity of computing and learning with multiplicative neural networks. *Neural Computation*, 14:241–301, 2002.
- D. J. Spiegelhalter and R. P. Knill-Jones. Statistical and knowledge-based approaches to clinical decision support systems. *Journal of the Royal Statistical Society, Series A*, 147:35–77, 1984.
- Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In Peter Auer and Ron Meir, editors, *Proceedings of the 18th Annual Conference on Learning Theory COLT 2005*, volume 3559 of *Lecture Notes in Artificial Intelligence*, pages 545–560. Springer-Verlag, Berlin, 2005.
- Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, Cambridge, MA, 2004.
- Koji Tsuda, Shotaro Akaho, Motoaki Kawanabe, and Klaus-Robert Müller. Asymptotic properties of the Fisher kernel. *Neural Computation*, 16:115–137, 2004.
- Koji Tsuda and Motoaki Kawanabe. The leave-one-out kernel. In Jose R. Dorronsoro, editor, *Proceedings of the International Conference on Artificial Neural Networks ICANN 2002*, volume 2415 of *Lecture Notes in Computer Science*, pages 727–732. Springer-Verlag, Berlin, 2002.
- Koji Tsuda, Motoaki Kawanabe, Gunnar Rätsch, Sören Sonnenburg, and Klaus-Robert Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14:2397–2414, 2002.
- Vladimir Vapnik. *Statistical Learning Theory*. Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley & Sons, New York, NY, 1998.
- Manfred K. Warmuth and S. V. N. Vishwanathan. Leaving the span. In Peter Auer and Ron Meir, editors, *Proceedings of the 18th Annual Conference on Learning Theory COLT 2005*, volume 3559 of *Lecture Notes in Artificial Intelligence*, pages 366–381. Springer-Verlag, Berlin, 2005.

Maximum Margin Algorithms with Boolean Kernels

Roni Khardon

*Department of Computer Science
Tufts University
Medford, MA 02155, USA*

RONI@CS.TUFTS.EDU

Rocco A. Servedio

*Department of Computer Science
Columbia University
New York, NY 10027, USA*

ROCCO@CS.COLUMBIA.EDU

Editor: Peter Bartlett

Abstract

Recent work has introduced Boolean kernels with which one can learn linear threshold functions over a feature space containing all conjunctions of length up to k (for any $1 \leq k \leq n$) over the original n Boolean features in the input space. This motivates the question of whether maximum margin algorithms such as Support Vector Machines can learn Disjunctive Normal Form expressions in the Probably Approximately Correct (PAC) learning model by using this kernel. We study this question, as well as a variant in which structural risk minimization (SRM) is performed where the class hierarchy is taken over the length of conjunctions.

We show that maximum margin algorithms using the Boolean kernels do not PAC learn $t(n)$ -term DNF for any $t(n) = \omega(1)$, even when used with such a SRM scheme. We also consider PAC learning under the uniform distribution and show that if the kernel uses conjunctions of length $\tilde{\omega}(\sqrt{n})$ then the maximum margin hypothesis will fail on the uniform distribution as well. Our results concretely illustrate that margin based algorithms may overfit when learning simple target functions with natural kernels.

Keywords: computational learning theory, kernel methods, PAC learning, Boolean functions

1. Introduction

Maximum margin algorithms, notably the Support Vector Machines (SVM) introduced by Boser et al. (1992), have received considerable attention in recent years (see, e.g., Shawe-Taylor and Cristianini, 2000, for an introduction). In their basic form, SVM learn linear threshold hypotheses and combine two powerful ideas. The first idea is to learn using the linear separator which achieves the *maximum margin* on the training data rather than an arbitrary consistent linear threshold hypothesis. The second idea is to use an implicit feature expansion by a *kernel function*. The kernel $K : X \times X \rightarrow \mathbb{R}$, where X is the original space of examples, computes the inner product in the expanded feature space. Given a kernel K which corresponds to some expanded feature space, the SVM hypothesis h is (an implicit representation of) the maximum margin linear threshold hypothesis over this expanded feature space rather than the original feature space. SVM theory (see, e.g., Shawe-Taylor and Cristianini, 2000) implies that if the kernel K is efficiently computable then it is possible to efficiently construct this maximum margin hypothesis h and that h itself is efficiently

computable. Several on-line algorithms have also been proposed which iteratively construct large margin hypotheses in the feature space (see, e.g., Friess et al., 1998; Gentile, 2001).

Both theoretical and experimental studies suggest that such algorithms may be able to take advantage of properties of the distribution and data to converge faster than what would be required by uniform convergence bounds. In particular, convergence bounds based on the maximum margin of the classifier on the observed data have been obtained by Shawe-Taylor et al. (1998) and by Shawe-Taylor and Cristianini (2000).

1.1 Can SVMs Learn DNF?

Another major focus of research in learning theory is the question of whether various classes of Boolean functions can be learned by computationally efficient algorithms. The canonical open question in this area is whether there exist efficient algorithms in the Probably Approximately Correct (PAC) learning model of Valiant (1984) for learning Boolean formulas in Disjunctive Normal Form, or DNF. This question has been open since the introduction of the PAC model and has been intensively studied by many researchers (see, e.g., Blum et al., 1994; Blum and Rudich, 1995; Bshouty, 1996; Hancock and Mansour, 1991; Jackson, 1997; Khardon, 1994; Klivans and Servedio, 2001; Kucera et al., 1994; Kushilevitz and Roth, 1993; Sakai and Maruoka, 2000; Tarui and Tsukiji, 1999; Verbeurgt, 1990, 1998).

In this paper we analyze the performance of maximum margin algorithms when used with Boolean kernels to learn DNF formulas. Several authors including Khardon et al. (2002), Sadohara (2001), Watkins (1999) and Kowalczyk et al. (2002) have recently proposed a family of kernel functions $K_k : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{N}$, where $1 \leq k \leq n$, such that $K_k(x, y)$ computes the number of (monotone or unrestricted) conjunctions of length (exactly or up to) k which are true in both x and y . This is equivalent to expanding the original feature space of n Boolean features to include all such conjunctions.¹ Since linear threshold elements can represent disjunctions, one can naturally view any DNF formula as a linear threshold function over this expanded feature space. It is thus natural to ask whether the K_k kernel maximum margin learning algorithms are good algorithms for learning DNF.

Additional motivation for studying DNF learnability with the K_k kernels comes from recent progress on the DNF learning problem. The fastest known algorithm for PAC learning DNF is due to Klivans and Servedio (2001); it works by explicitly expanding each example into a feature space of monotone conjunctions and explicitly learning a consistent linear threshold function over this expanded feature space. Since the K_k kernel enables us to do such expansions implicitly in a computationally efficient way, it is natural to investigate whether the K_k -kernel maximum margin algorithm yields a computationally efficient algorithm for PAC learning DNF.

1.2 Discussion of the Problem and Previous Work

Recall that a polynomial size sample is sufficient for PAC learning any concept class where each concept in the class has a polynomial size description. In any such case, as shown by Blumer et al. (1987), an Occam algorithm which identifies a short consistent hypothesis in the class is a

1. This Boolean kernel is similar to the well known polynomial kernel in that all monomials of length up to k are represented. The main difference is that the polynomial kernel assigns weights to monomials which depend on certain binomial coefficients; thus the weights of different monomials can differ by an exponential factor. In the Boolean kernel all monomials have the same weight.

PAC learner. Thus the statistical ingredient of the problem of PAC learning polynomial size DNF expressions is in some sense solved and the main question seems to be computational. Yet, it is not known whether such an Occam algorithm exists.

As mentioned above recent work of Shawe-Taylor et al. (1998) and Shawe-Taylor and Cristianini (2000) has introduced convergence bounds for maximum margin learners. These bounds are independent of the dimension of the expanded feature space but they depend on the L_2 norm of examples in this space, as well as the margin obtained on the sample. In particular they depend on R/δ where δ is the margin and R bounds the L_2 norm of examples. It is instructive to consider applying these results in our setting, where we assume for concreteness that we are learning a function given by one k -monomial T , and that we are using the K_k monotone kernel with the maximum margin algorithm. The linear threshold representation for this function is $x_T \geq 1$, i.e. only one weight is non-zero and the (non-normalized) margin obtained is 1. However, the maximum L_2 norm of examples is $\Theta(n^{k/2})$ so the quantity R/δ is exponentially large. Seen in another way, we can normalize the examples to have a maximum norm of 1, but then the normalized margin obtained is $\Theta(n^{-k/2})$. Indeed, the bound given by Theorem 4.18 of the paper of Shawe-Taylor and Cristianini (2000) only implies nontrivial generalization error for the K_k kernel algorithm if a sample of size $n^{\Omega(k)}$ is used, and with such a large sample the computational advantage of using the K_k kernel is lost. As a result, using such bounds we cannot *a priori* conclude anything about the performance of the algorithm when it is run with a polynomial size sample.

Recently, several negative results have been obtained for embedding concept classes into Euclidean spaces (Ben-David et al., 2002; Forster et al., 2003). The results are best understood in terms of their relation to the convergence bounds. For example, Ben-David et al. (2002) show that there are concept classes for which there is no mapping into $[0, 1]^N$ that achieves a large margin, for any N . This actually holds “for the majority of concept classes with low VC dimension”. Other work of Forster et al. (2003) gives bounds on the margin (or the dimension required) for concrete concept classes. Again, the implication is that known convergence bounds do not imply success in these cases. It is worth noting that the notion of embedding used in these results is slightly stronger than the requirement in the upper bounds, in that the embedding and margin are for all the examples (or a large fraction of the instance space) and not just for a small sample. However, these results rule out any simple application of the upper bounds that use properties of the concept class directly.

Therefore, in many cases, and concretely in our case of learning DNF via the monomial kernel, the upper bounds provided by standard convergence theorems only imply that a large sample will guarantee successful generalization. However, such upper bounds do *not* imply that the K_k kernel maximum margin algorithm must have poor generalization error if run with a smaller sample. This is precisely the question studied in this paper. Notice the contrast with the discussion of Occam algorithms; here we have an efficient algorithm with no known bounds on hypothesis size. The question is whether its hypothesis provides a good generalization in a statistical sense.

The notion that this might succeed is not unreasonable. In an analogous situation, Servedio (1999) studied the generalization error of the Perceptron and Winnow algorithms for various problems. For both Perceptron and Winnow the standard bounds gave only an exponential upper bound on the number of examples required to learn various classes, but a detailed algorithm-specific analysis showed that the Perceptron algorithm succeeds in polynomial time whereas the Winnow algorithm requires exponential time for the problems considered. Analogously, in this paper we perform detailed algorithm-specific analysis for the K_k kernel maximum margin algorithms.

In previous work we have studied a similar question with regard to the perceptron algorithm. In particular, Khardon et al. (2002) constructed a simple Boolean function and an example sequence for the online mistake-bound learning model, and showed that this sequence causes the K_n kernel Perceptron algorithm (i.e. the Perceptron algorithm run over a feature space of all 2^n monotone conjunctions) to make $2^{\Omega(n)}$ many mistakes. The current paper differs in several ways from this earlier work: we study the maximum margin algorithm rather than Perceptron, we consider PAC learning from a random sample rather than online learning, and we analyze the K_k kernels for all $1 \leq k \leq n$. We note here that maximum margin linear threshold learning algorithms are generally viewed as being more powerful than the simple Perceptron algorithm, and that PAC learning is generally viewed as being easier than online mistake bound learning (it is well known that any concept class which is efficiently learnable in the mistake bound model is efficiently PAC learnable, but the converse is not true as shown by Blum, 1994). Thus, the results of this work represent a substantial strengthening and generalization of the work of Khardon et al. (2002).

1.3 Our Results

In this paper we study the kernels corresponding to all monotone monomials of length up to k , which we denote by K_k . We also consider the polynomial kernel $K(x, y) = (x \cdot y)^k$, parametrized by the degree of the polynomial.

In addition to unaugmented maximum margin algorithms we also consider a natural scheme of structural risk minimization (SRM) that can be used with maximum margin algorithms over this family of Boolean kernels. In SRM, given a hierarchy of classes $C_1 \subseteq C_2 \subseteq \dots$, one learns with each class separately and uses a cost function combining the complexity of the class with its observed accuracy to choose the final hypothesis. The cost function typically balances various criteria such as the observed error and the (bound on) generalization error. A natural scheme here is to use SRM over the classes formed by K_k with $k = 1, \dots, n$.²

Combining either of these algorithms (i.e. with or without SRM scheme) with the monomial kernel we get a concrete and efficient algorithm that can be applied to the problem of learning DNF. We prove several negative results which establish strong limitations on the ability of such algorithms to learn DNF. Similar negative results are proved for the polynomial kernel as well.

Our first result says essentially that for any $t(n) = \omega(1)$, for all $k = 1, \dots, n$ the K_k kernel maximum margin algorithm cannot PAC learn $t(n)$ -term DNF. More precisely, we prove

Result 1: Let $t(n) = \omega(1)$ and let $\varepsilon = \frac{1}{4 \cdot 2^{t(n)}}$. There is a $O(t(n)^{1/3})$ -term monotone DNF over $t(n)$ relevant variables, and a distribution \mathcal{D} over $\{0, 1\}^n$ such that for all $k \in \{1, \dots, n\}$ the K_k maximum margin hypothesis has error larger than ε (with overwhelmingly high probability over the choice of a polynomial size random sample from \mathcal{D}).

Note that this result implies that the K_k maximum margin algorithms fail even when combined with SRM *regardless of the cost function*. This is simply because the maximum margin hypothesis has error $> \varepsilon$ for all k , and hence the final SRM hypothesis must also have error $> \varepsilon$.

While our accuracy bound in the above result is small (it is $o(1)$ since $t(n) = \omega(1)$), a simple variant of the construction used for Result 1 also proves:

2. This is standard practice in experimental work with the polynomial kernel, where typically small values of k are tried (e.g. 1 to 5) and the best is chosen.

Result 2: Let $f(x) = x_1$ be the target function. There is a distribution \mathcal{D} over $\{0, 1\}^n$ such that for any $k = \omega(1)$ the K_k maximum margin hypothesis has error at least $\frac{1}{2} - 2^{-n^{\Omega(1)}}$ (with overwhelmingly high probability over the choice of a polynomial size random sample from \mathcal{D}).

Thus any attempt to learn using monomials of non-constant size can provably lead to overfitting. Note that for any $k = \Theta(1)$, standard bounds on maximum margin algorithms show that the K_k kernel algorithm can learn $f(x) = x_1$ from a polynomial size sample.

Given these strong negative results for PAC learning under arbitrary distributions, we next consider the problem of PAC learning monotone DNF under the uniform distribution. This is one of the few frameworks in which some positive results have been obtained for learning DNF from random examples only (see, e.g., Bshouty and Tamon, 1996; Servedio, 2001). In this scenario a simple variant of the construction for Result 1 shows that learning must fail if k is too small:

Result 3: Let $t(n) = \omega(1)$ and $\epsilon = \frac{1}{4 \cdot 2^{t(n)}}$. There is a $O(t(n)^{1/3})$ -term monotone DNF over $t(n)$ relevant variables such that for all $k < t(n)$ the K_k maximum margin hypothesis has error at least ϵ (with probability 1 over the choice of a random sample from the uniform distribution).

This result is representation based; we show that no possible hypothesis output by the K_k algorithm can have error less than ϵ . On the other hand, we also show that the K_k algorithm fails under the uniform distribution for large k :

Result 4: Let $f(x) = x_1$ be the target function. For any $k = \tilde{\omega}(\sqrt{n})$, the K_k maximum margin hypothesis will have error $\frac{1}{2} - 2^{-\Omega(n)}$ with probability at least 0.028 over the choice of a polynomial size random sample from the uniform distribution.

Note that there is a substantial gap between the “low” values of k (for which learning is guaranteed to fail) and the “high” values of k (for which we show that learning fails with constant probability). It is of significant interest to characterize the performance of the K_k maximum margin algorithm under the uniform distribution for these intermediate values of k ; a discussion of this point is given in Section 5.

2. Preliminaries

We consider learning Boolean functions over the Boolean cube $\{0, 1\}^n$ so that $f : \{0, 1\}^n \rightarrow \{0, 1\}$. It is convenient to consider instead the range $\{-1, 1\}$ with 0 mapped to -1 and 1 mapped to 1. This is easily achieved by the transformation $f'(x) = 1 - 2f(x)$ and since we deal with linear function representations this can be done without affecting the results. For the rest of the paper we assume this representation.

For $x, y \in \mathbb{R}^n$ we write $x \cdot y$ to denote the standard inner product $\sum_{i=1}^n x_i y_i$. Note that for $x, y \in \{0, 1\}^n$, $x \cdot y$ calculates the number of bits which are 1 in both x and y . Our arguments will refer to L_1 and L_2 norms of vectors for which we use the notation $|x| = \sum |x_i|$ and $\|x\| = \sqrt{\sum x_i^2}$.

Definition 1 Let $h : \mathbb{R}^N \rightarrow \{-1, 1\}$ be a linear threshold function $h(x) = \text{sign}(W \cdot x - \theta)$ for some $W \in \mathbb{R}^N, \theta \in \mathbb{R}$. The margin of h on $\langle z, b \rangle \in \mathbb{R}^N \times \{-1, 1\}$ is

$$m_h(z, b) = \frac{b(W \cdot z - \theta)}{\|W\|}.$$

Note that $|m_h(z, b)|$ is the Euclidean distance from z to the hyperplane $W \cdot x = \theta$.

Definition 2 Let $S = \{\langle x^i, b_i \rangle\}_{i=1, \dots, m}$ be a set of labeled examples where each $x^i \in \mathbb{R}^N$ and each $b_i \in \{-1, 1\}$. Let $h(x) = \text{sign}(W \cdot x - \theta)$ be a linear threshold function. The margin of h on S is

$$m_h(S) = \min_{\langle x, b \rangle \in S} m_h(x, b).$$

The maximum margin classifier for S is the linear threshold function $h(x) = \text{sign}(W \cdot x - \theta)$ such that

$$m_h(S) = \max_{W' \in \mathbb{R}^N, \theta' \in \mathbb{R}} \min_{\langle x, b \rangle \in S} \frac{b(W' \cdot x - \theta')}{\|W'\|}. \quad (1)$$

The quantity (1) is called the margin of S and is denoted m_S .

Note that $m_S > 0$ iff S is consistent with some linear threshold function. If $m_S > 0$ then the maximum margin classifier for S is unique (see, e.g., Shawe-Taylor and Cristianini, 2000).

For a sample S and example $\langle x^i, 1 \rangle$ in S we sometimes write $x^{i,+}$ to indicate that x^i is a positive example. Similarly $x^{j,-}$ is used to indicate that x^j is a negative example.

Let ϕ be a transformation which maps $\{0, 1\}^n$ to \mathbb{R}^N and let $K : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{R}$ be the corresponding kernel function $K(x, y) = \phi(x) \cdot \phi(y)$. Given a set of labeled examples $S = \{\langle x^i, b_i \rangle\}_{i=1, \dots, m}$ where each x^i belongs to $\{0, 1\}^n$ we denote by $\phi(S)$ the set of transformed examples $\{\langle \phi(x^i), b_i \rangle\}_{i=1, \dots, m}$.

We refer to the following learning algorithm as the *K-maximum margin learner*:

- The algorithm takes as input a sample $S = \{\langle x^i, b_i \rangle\}_{i=1, \dots, m}$ of m labeled examples.

We assume that S contains both positive and negative examples, and that the sample is linearly separable. If these conditions do not hold then the maximum margin hypothesis is not defined. The assumptions are simply used to rule out the degenerate cases from the analysis.

We also assume that $m = \text{poly}(n)$, i.e. that $m = n^{\Theta(1)}$ and we have both a lower and upper bound on the number of examples. The upper bound as usual limits the resources the algorithm uses. The lower bound is again simply used to rule out degenerate cases from the analysis.

- The algorithm's hypothesis is $h : \{0, 1\}^n \rightarrow \{-1, 1\}$, $h(x) = \text{sign}(W \cdot \phi(x) - \theta)$ where $\text{sign}(W \cdot x - \theta)$ is the maximum margin classifier for $\phi(S)$. Without loss of generality we assume that W is normalized, that is $\|W\| = 1$.

SVM theory tells us that if $K(x, y)$ can be computed in $\text{poly}(n)$ time then the K -maximum margin learning algorithm runs in $\text{poly}(n, m) = \text{poly}(n)$ time and the output hypothesis $h(x)$ can be evaluated in $\text{poly}(n, m) = \text{poly}(n)$ time (see, e.g., Shawe-Taylor and Cristianini, 2000).

Our goal is to analyze the PAC learning ability of various kernel maximum margin learning algorithms. Recall (see, e.g., Kearns and Vazirani, 1994) that a PAC learning algorithm for a class C of functions over $\{0, 1\}^n$ is an algorithm which runs in time polynomial in n and $\frac{1}{\delta}$, $\frac{1}{\epsilon}$ where δ is a confidence parameter and ϵ is an accuracy parameter. We assume here, as is the case throughout the paper, that each function in C has a description of size $\text{poly}(n)$. Given access to random labeled examples $\langle x, f(x) \rangle$ for any $f \in C$ and any distribution \mathcal{D} over $\{0, 1\}^n$, with probability at least $1 - \delta$ a PAC learning algorithm must output an efficiently computable hypothesis h such that $\Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$. Applying this framework to the maximum margin learner, we assume that the sample S is drawn by taking IID samples from \mathcal{D} and providing the label according to the target function f . If

an algorithm only satisfies this criterion for a particular distribution such as the uniform distribution on $\{0, 1\}^n$, we say that it is a uniform distribution PAC learning algorithm.

Let $\rho_k(n) = \sum_{i=1}^k \binom{n}{i}$. Note that the number of nonempty monotone conjunctions (i.e. monomials) of size at most k on n variables is $\rho_k(n)$. For $x \in \{0, 1\}^n$ we write $\phi_k(x)$ to denote the $\rho_k(n)$ -dimensional vector $(x_T)_{T \subseteq \{1, \dots, n\}, 1 \leq |T| \leq k}$ where $x_T = \prod_{i \in T} x_i$, i.e. the components of $\phi_k(x)$ are all monotone conjunctions of the desired size. We note that for an example $x \in \{0, 1\}^n$, the L_1 norm of the expanded example $\phi_k(x)$ is $|\phi_k(x)| = \rho_k(|x|)$.

Definition 3 We write $K_k(x, y)$ to denote $\phi_k(x) \cdot \phi_k(y)$. We refer to K_k as the k -monomials kernel.

The following theorem shows that the k -monomial kernels are easy to compute:

Theorem 4 (Khardon, Roth, and Servedio, 2002) For all $1 \leq k \leq n$ we have $K_k(x, y) = \sum_{i=1}^k \binom{x \cdot y}{i}$.

We will frequently use the following observation which is a direct consequence of the Cauchy-Schwarz inequality:

Observation 1 If $U \in \mathbb{R}^{N_1}$ with $\|U\| = L$ and $I \subseteq \{1, \dots, N_1\}$, $|I| = N_2$, then $\sum_{i \in I} |U_i| \leq L \cdot \sqrt{N_2}$.

As a consequence of Observation 1 we have that if $\rho_k(n) = N_1$ is the number of features in the expanded feature space and $|\phi_k(x)| = \rho_k(|x|) = N_2$, then $U \cdot \phi_k(x) \leq L \cdot \sqrt{N_2}$.

Finally we also use the following well-known tail bound on sums of independent random variables (see, e.g., Kearns and Vazirani, 1994):

Fact 2 (Chernoff Bounds) Let X_1, \dots, X_m be a sequence of m independent 0/1-valued random variables, each of which has $E[X_i] = p$. Let X denote $\sum_{i=1}^m X_i$, so $E[X] = pm$. Then for $0 \leq \gamma \leq 1$, we have

$$\Pr[X > (1 + \gamma)pm] \leq e^{-mp\gamma^2/3} \quad \text{and} \quad \Pr[X < (1 - \gamma)pm] \leq e^{-mp\gamma^2/2}.$$

3. Distribution-Free Non-Learnability

We give a DNF and a distribution which are such that the maximum margin algorithm using the k -monomials kernel fails to learn, for all $1 \leq k \leq n$. The DNF we consider is a read once monotone DNF over $t(n)$ variables where $t(n) = \omega(1)$ and $t(n) = O(\log n)$. In fact our results hold for any $t(n) = \omega(1)$ but for concreteness we use $t(n) = \log n$ as a running example. Let

$$f(x) = (x_1 \cdots x_{4\ell^2}) \vee (x_{4\ell^2+1} \cdots x_{8\ell^2}) \vee \cdots \vee (x_{4\ell^3-4\ell^2+1} \cdots x_{4\ell^3}) \tag{2}$$

where $4\ell^3 = t(n) = \log n$ so that the number of terms ℓ equals $\Theta(t(n)^{1/3}) = \Theta((\log n)^{1/3})$. For the rest of this section $f(x)$ will refer to the function defined in Equation (2) and ℓ to its size parameter.

A *polynomial threshold function* is defined by a multivariate polynomial $p(x_1, \dots, x_n)$ with real coefficients. The output of the polynomial threshold function is 1 if $p(x_1, \dots, x_n) \geq 0$ and is -1 otherwise. The degree of the function is the degree of the polynomial p . A simple but useful observation is that any hypothesis output by the K_k kernel maximum margin algorithm must be a polynomial threshold function of degree at most k . Minsky and Papert (1968) (see also Klivans and Servedio, 2001) gave the following lower bound on polynomial threshold function degree for DNF:

Theorem 5 Any polynomial threshold function for $f(x)$ in Equation (2) must have degree at least ℓ .

The distribution \mathcal{D} on $\{0, 1\}^n$ we consider is the following:

- With probability $\frac{1}{2}$ the distribution outputs 0^n .
- With probability $\frac{1}{2}$ the distribution outputs a string $x \in \{0, 1\}^n$ drawn from the following product distribution \mathcal{D}' : the first $t(n)$ bits are drawn uniformly, and the last $n - t(n)$ bits are drawn from the product distribution which assigns 1 to each bit with probability $\frac{1}{n^{1/3}}$.

For small values of k the result is representation based and does not depend on the sample drawn:

Lemma 6 If the maximum margin algorithm uses the kernel K_k for $k < \ell$ when learning $f(x)$ under \mathcal{D} then its hypothesis has error greater than $\varepsilon = \frac{1}{4 \cdot 2^{t(n)}} = \frac{1}{4n}$.

Proof If hypothesis h has error at most $\varepsilon = \frac{1}{4 \cdot 2^{t(n)}}$ under \mathcal{D} then clearly it must have error at most $\frac{1}{2 \cdot 2^{t(n)}}$ under \mathcal{D}' . Since we are using the kernel K_k , the hypothesis h is some polynomial threshold function of degree at most k which has error $\tau \leq \frac{1}{2 \cdot 2^{t(n)}}$ under \mathcal{D}' . So there must be some setting of the last $n - t(n)$ variables which causes h to have error at most τ under the uniform distribution on the first $t(n)$ bits. Under this setting of variables the hypothesis is a degree- k polynomial threshold function on the first $t(n)$ variables. By Minsky and Papert’s theorem, this polynomial threshold function cannot compute the target function exactly, so it must be wrong on at least one setting of the first $t(n)$ variables. But under the uniform distribution, every setting of those variables has probability at least $\frac{1}{2^{t(n)}}$. This contradicts $\tau \leq \frac{1}{2 \cdot 2^{t(n)}}$. ■

For larger values of k (in fact for all $k = \omega(1)$) we show that with high probability the maximum margin hypothesis will overfit the sample. We start by explaining the high level structure of the proof. Note that the target function depends on a small number of the features so most features are irrelevant for the target. On the other hand the distribution is constructed such that each example in the sample has a “large” weight on its own, whereas the weight of the common features in any two examples is “small”. As a result of these facts, one can find a simple hypothesis with relatively large margin by using all the structure from the examples, i.e. fitting them exactly. Naturally such a hypothesis overfits the sample and provides little by way of generalizing to other examples. It is hard in general to analyze the maximum margin hypothesis directly, and in particular it does not necessarily follow the overfitting scheme of the simple hypothesis. However, our analysis uses the simple hypothesis to infer some properties of the maximum margin hypothesis and through this provide error bounds for it. The same structure is used again to analyze the polynomial kernel and for the analysis of the uniform distribution. However, the technical details underlying the analysis are different in each case.

The following definition captures typical properties of a sample from distribution \mathcal{D} :

Definition 7 A sample S is a \mathcal{D} -typical sample if

- The sample includes the example 0^n .
- Any nonzero example x in the sample has $0.99n^{2/3} \leq |x| \leq 1.01n^{2/3}$.

- Every pair of examples $x^{i,+}$ and $x^{j,-}$ in S satisfies $x^{i,+} \cdot x^{j,-} \leq 1.01n^{1/3}$.

We can apply Chernoff bounds to analyze the second and third conditions in the definition (with $p = \frac{1}{n^{1/3}}$ and $p = \frac{1}{n^{2/3}}$ respectively) over the last $n - t(n) > n/2$ bits, and absorb the first $t(n)$ bits in the multiplicative (1 ± 0.01) divergence from the expected value in each case (recall that $t(n)$ is only $O(\log n)$). We thus have that the second and third conditions each fail with probability at most $2^{-n^{\Omega(1)}}$. Since the maximum margin algorithm uses $m = \text{poly}(n) = n^{\Omega(1)}$ many examples (see Section 2), the first condition fails with probability $2^{-m} = 2^{-n^{\Omega(1)}}$ as well. A union bound thus gives:

Lemma 8 For $m = \text{poly}(n)$, with probability $1 - 2^{-n^{\Omega(1)}}$ a random i.i.d. sample of m draws from \mathcal{D} is a \mathcal{D} -typical sample.

Definition 9 Let S be a sample. The set $Z(S)$ consists of all positive examples $z \in \{0, 1\}^n$ (i.e. $f(z) = 1$) which have the property that every example x in S satisfies $x \cdot z \leq 1.01n^{1/3}$.

As above, we can apply Chernoff bounds with $p = \frac{1}{n^{2/3}}$ and use the union bound over all examples $x \in S$ to show that the probability that a random example z drawn from \mathcal{D} will have $x \cdot z > 1.01n^{1/3}$ for any $x \in S$ is at most $2^{-n^{\Omega(1)}}$. Recall that f only depends on the first $t(n)$ bits and its terms are shorter than $t(n)$. Since the distribution is uniform over these bits we have $\Pr[f(z) = 1] \geq \frac{1}{2^{t(n)}} = \frac{1}{n}$. Thus, conditioning on z being a positive example we still have:

Lemma 10 Let S be a \mathcal{D} -typical sample of size $m = \text{poly}(n)$ examples. Then $\Pr_{\mathcal{D}}[z \in Z(S) | f(z) = 1] = 1 - 2^{-n^{\Omega(1)}}$.

We now show that for a \mathcal{D} -typical sample one can achieve a very large margin:

Lemma 11 Let S be a \mathcal{D} -typical sample. Then the maximum margin m_S satisfies

$$m_S \geq M_{h'} \equiv \frac{1}{2} \cdot \frac{\rho_k(.99n^{2/3}) - m\rho_k(1.01n^{1/3})}{\sqrt{m\rho_k(1.01n^{2/3})}}.$$

Proof We exhibit an explicit linear threshold function h' which has margin at least $M_{h'}$ on the data set. Let $h'(x) = \text{sign}(W' \cdot \phi(x) - \theta')$ be defined as follows:

- $W'_T = 1$ if T is satisfied in some positive example;
- $W'_T = 0$ if T is not satisfied in any positive example.
- θ' is the value that gives the maximum margin on $\phi_k(S)$ for this W' , i.e. θ' is the average of the smallest value of $W' \cdot \phi_k(x^{i,+})$ and the largest value of $W' \cdot \phi_k(x^{j,-})$.

Since each positive example x^+ in S has at least $.99n^{2/3}$ ones, we have $W' \cdot \phi(x^+) \geq \rho_k(.99n^{2/3})$. Since each positive example has at most $1.01n^{2/3}$ ones, each positive example in the sample contributes at most $\rho_k(1.01n^{2/3})$ ones to W' , so $\|W'\| \leq \sqrt{m\rho_k(1.01n^{2/3})}$.

Finally, for any negative example x^- in the sample a term T contributes to $W' \cdot \phi(x^-)$ only if T is true in x^- and in some positive example. Now since x^- shares at most $1.01n^{1/3}$ ones with any positive example in the sample, the number of such terms is at most $m\rho_k(1.01n^{1/3})$. We therefore

get $W' \cdot \phi(x^-) \leq m\rho_k(1.01n^{1/3})$. Putting these conditions together, we get that the margin of h' on the sample is at least

$$\frac{1}{2} \cdot \frac{\rho_k(.99n^{2/3}) - m\rho_k(1.01n^{1/3})}{\sqrt{m\rho_k(1.01n^{2/3})}}$$

as desired. ■

It is instructive to use a rough calculation and compare the margin obtained to the one calculated in the introduction. The main term in the bound above grows roughly as $\sqrt{\frac{\rho_k(n^{2/3})}{m}}$ which is exponentially larger than the constant value obtained by the correct classifier.

Lemma 12 *If S is a \mathcal{D} -typical sample, then the threshold θ in the maximum margin classifier for S is at least $M_{h'}$.*

Proof Let $h(x) = \text{sign}(W \cdot \phi(x) - \theta)$ be the maximum margin hypothesis. Since $\|W\| = 1$ we have

$$\theta = \frac{\theta}{\|W\|} = m_h(\phi_k(0^n), -1) \geq m_{h'}(S) \geq M_{h'}$$

where the second equality holds because $W \cdot \phi(0^n) = 0$ and the last inequality is by Lemma 11. ■

Lemma 13 *If the maximum margin algorithm uses the kernel K_k for $k = \omega(1)$ when learning $f(x)$ under \mathcal{D} then with probability $1 - 2^{-n^{\Omega(1)}}$ its hypothesis has error greater than $\varepsilon = \frac{1}{4 \cdot 2^{f(n)}} = \frac{1}{4n}$.*

Proof Let S be the sample used for learning and let $h(x) = \text{sign}(W \cdot \phi_k(x) - \theta)$ be the maximum margin hypothesis. It is well known (see, e.g., Shawe-Taylor and Cristianini, 2000, Proposition 6.5) that the maximum margin weight vector W is a linear combination of the support vectors, i.e. of certain examples $\phi_k(x)$ in the sample $\phi_k(S)$. Hence the only coordinates W_T of W that can be nonzero are those corresponding to features (conjunctions) T such that $x_T = 1$ for some example x in S .

By Lemma 8 we have that with probability $1 - 2^{-n^{\Omega(1)}}$ the sample S is \mathcal{D} -typical. Consider any $z \in Z(S)$. It follows from the above observations on W that $W \cdot \phi_k(z)$ is a sum of at most $m\rho_k(1.01n^{1/3})$ nonzero numbers, and moreover the sum of the squares of these numbers is at most 1. Thus by Observation 1 we have that $W \cdot \phi_k(z) \leq \sqrt{m\rho_k(1.01n^{1/3})}$. The positive example z is erroneously classified as negative by h if $\theta > W \cdot \phi_k(z)$; by Lemma 12 this inequality holds if

$$\frac{1}{2} \cdot \frac{\rho_k(.99n^{2/3}) - m\rho_k(1.01n^{1/3})}{\sqrt{m\rho_k(1.01n^{2/3})}} > \sqrt{m\rho_k(1.01n^{1/3})},$$

i.e. if

$$\rho_k(.99n^{2/3}) > 2m\sqrt{\rho_k(1.01n^{1/3})\rho_k(1.01n^{2/3})} + m\rho_k(1.01n^{1/3}). \quad (3)$$

We prove in Appendix A that this holds for any $k = \omega(1)$.

Finally, observe that positive examples have probability at least $\frac{1}{2^{f(n)}} = \frac{1}{n}$. The above argument shows that any $z \in Z(S)$ is misclassified, and Lemma 10 guarantees that the relative weight of $Z(S)$ in positive examples is $1 - 2^{-n^{\Omega(1)}}$. Thus the overall error rate of h under \mathcal{D} is at least

$(1 - 2^{-n^{\Omega(1)}}) \frac{1}{2^{f(n)}} > \frac{1}{4 \cdot 2^{f(n)}} = \frac{1}{4n}$ as claimed. ■

Together, Lemma 6 and Lemma 13 imply Result 1:

Theorem 14 *For any value of k , if the maximum margin algorithm (as defined in Section 2) uses the kernel K_k when learning $f(x)$ under \mathcal{D} then with probability $1 - 2^{-n^{\Omega(1)}}$ its hypothesis has error greater than $\varepsilon = \frac{1}{4 \cdot 2^{f(n)}} = \frac{1}{4n}$.*

With a small modification we can also obtain Result 2. In particular, since we do not need to deal with small k we can use a simple function $f = x_1$ and modify \mathcal{D} as follows. With probability $\frac{1}{4}$ the assignment 0^n is drawn. With probability $\frac{3}{4}$ we draw from \mathcal{D}' where $x_1 = 1$ with probability $\frac{2}{3}$ and as before the other bits are 1 with probability $\frac{1}{n^{1/3}}$. Note that for the modified distribution the probability that $f(x) = 1$ is 0.5. It is easy to see that the previous arguments go through for this case and we get:

Theorem 15 *For $k = \omega(1)$, if the maximum margin algorithm uses the kernel K_k when learning $f(x) = x_1$ under \mathcal{D} then with probability $1 - 2^{-n^{\Omega(1)}}$ its hypothesis has error at least $\varepsilon = \frac{1}{2} - 2^{-n^{\Omega(1)}}$.*

Remark 16 The proofs above can be adapted to show the same non-learnability results for the polynomial kernel $K_k(x, y) = (x \cdot y)^k$ which is commonly being used with SVM systems. The low degree argument in Lemma 6 holds directly. We briefly sketch the ideas for the high degree case. First note that Lemmas 8 and 10 hold without modification. The argument in Lemma 11 does not go through if we use the same value of W' (since W' is defined in the expanded feature space and $\phi(x)$ is not a zero-one vector, it is not as easy to argue about the value of $W' \cdot \phi(x)$). However, we can use a simple modification to get a similar result. First note that for any $x \in \{0, 1\}^n$, all features in $\phi(x)$ take only non-negative values. Now define W' to be $W' = \sum_{x^{j,+} \in S} \phi(x^{j,+})$. As in Lemma 11 we have:

- $W' \cdot \phi(x^+) = \sum_{x^{j,+} \in S} \phi(x^{j,+}) \cdot \phi(x^+) \geq \phi(x^+) \cdot \phi(x^+) \geq (0.99n^{2/3})^k$ where the first inequality uses the fact that all features in the expanded space have a positive value and therefore all inner products in the sum are positive.
- $W' \cdot \phi(x^-) = \sum_{x^{j,+} \in S} \phi(x^{j,+}) \cdot \phi(x^-) \leq m(1.01n^{1/3})^k$.
- $\|W'\| = \sqrt{(\sum_{x^{j,+} \in S} \phi(x^{j,+})) \cdot (\sum_{x^{j,+} \in S} \phi(x^{j,+}))} \leq \sqrt{m^2(1.01n^{2/3})^k}$.

So the maximum margin is at least

$$\frac{1}{2} \cdot \frac{(.99n^{2/3})^k - m(1.01n^{1/3})^k}{m\sqrt{(1.01n^{2/3})^k}}. \quad (4)$$

Now the proof of Lemma 12 shows that (4) is a lower bound on the threshold of the maximum margin classifier.

The argument in Lemma 13 needs to be changed since we need a bound on $W \cdot \phi(z)$. This can be derived as follows. Let U be such that $U_i \geq 0$ and $U_i = |W_i|$ so weights in U and W have the same

magnitude but the weights in U are forced to be non-negative. Then we have that $\|U\| = \|W\| = 1$. For an example $z \in Z(S)$ we now have

$$\begin{aligned} W \cdot \phi(z) &\leq U \cdot \phi(z) \\ &\leq \sum_{x^i \in S} U \cdot \phi(z \cap x^i) \\ &\leq m(1.01n^{1/3})^{k/2}. \end{aligned}$$

The first inequality holds since all entries in $\phi(z)$ are non-negative. The second inequality is true since both vectors do not have negative weights and a monomial contributes to $W \cdot \phi(z)$ only if it is true both in z and in at least one example in the sample (recall that, as in the proof of Lemma 13, the vector W is a linear combination of vectors $\phi(x) \in \phi(S)$). Therefore, each weight in $\phi(z)$ is represented by a weight in one of the intersections, and the value of the weight depends only on the monomial so it is the same in $\phi(z)$ and $\phi(z \cap x^i)$. Summing over all x_i in S gives an upper bound on the total contribution to $W \cdot \phi(z)$. The last inequality follows from the Cauchy-Schwarz inequality.

As a result of this upper bound on $W \cdot \phi(z)$, we have that z is misclassified if

$$\frac{1}{2} \cdot \frac{(.99n^{2/3})^k - m(1.01n^{1/3})^k}{m\sqrt{(1.01n^{2/3})^k}} > m\sqrt{(1.01n^{1/3})^k}.$$

This can be shown to hold for all $k = \omega(1)$.

4. Uniform Distribution

While Theorem 14 tells us that the K_k -maximum margin learner is not a PAC learning algorithm for monotone DNF in the distribution-free PAC model, it does not rule out the possibility that the K_k -maximum margin learner might succeed for particular probability distributions such as the uniform distribution on $\{0, 1\}^n$. In this section we investigate the uniform distribution.

It is easy to observe that the proof of Lemma 6 goes through for the uniform distribution as well (we actually gain a factor of 2). This therefore proves Result 3: if the algorithm uses too low a degree k then its hypothesis cannot possibly be a sufficiently accurate approximation of the target. In contrast, the next result will show that if a rather large k is used then the algorithm is likely to overfit.

The case of large k is more complex. In Section 3 we took advantage of the fact that 0^n occurred with high weight under the distribution \mathcal{D} . This provided a lower bound (of 0) on the value of $W \cdot \phi_k(x)$ for some negative example in the sample, and then we could argue that the value of θ in the maximum margin classifier must be at least as large as m_S . For the uniform distribution, though, this lower bound no longer holds, so we must use a more subtle analysis. Before explaining the idea we need some technical details.

For the next result, we consider the target function $f(x) = x_1$. Let $S = S^+ \cup S^-$ be a data set drawn from the uniform distribution \mathcal{U} and labeled according to the function $f(x)$ where $S^+ = \{x^{i,+}, 1\}_{i=1, \dots, m_+}$ are the positive examples and $S^- = \{x^{j,-}, -1\}_{j=1, \dots, m_-}$ are the negative examples. Let u_i denote $|x^{i,+}|$ the weight of the i -th positive example, and let the positive examples be ordered so that $u_1 \leq u_2 \leq \dots \leq u_{m_+}$. Similarly let v_j denote $|x^{j,-}|$ the weight of the j -th negative example with $v_1 \leq v_2 \leq \dots \leq v_{m_-}$.

It turns out that the relative sizes of u_1 and v_1 , the weights of the lightest positive and negative examples in S , play an important role. This is captured by the following definition:

Definition 17 A sample S of size m is positive-skewed if $u_1 \geq v_1 + B$, i.e. the lightest positive example in S weighs at least B more than the lightest negative example, where $B = \frac{1}{66} \sqrt{\frac{n}{\log m}}$.

Now, if the sample is positive skewed we can calculate a lower bound on $W \cdot \phi_k(x)$ for negative examples in the sample. The value of B is chosen so that this bound can be used to give a non-trivial bound for θ . The details of this argument are developed in Section 4.2. But we must first establish that the algorithm may indeed get a positive-skewed sample as input.

4.1 The Probability of Obtaining Positive-Skewed Samples

Theorem 18 Let S be a sample of size $m = \text{poly}(n)$ drawn from the uniform distribution. Then S is positive-skewed with probability at least 0.029.

Proof Our first step is to reduce to a situation in which the positive examples and negative examples are independent from each other.³

Let M_-, M_+ be any two positive integers. Consider the following new probabilistic experiment which we call E_{M_-, M_+} : first M_- draws are made from a binomial distribution $B(n-1, \frac{1}{2})$ to obtain (sorted) values $v_1 \leq \dots \leq v_{M_-}$, and then M_+ draws are made from $1 + B(n-1, \frac{1}{2})$ to obtain (sorted) values $u_1 \leq \dots \leq u_{M_+}$. The values v_1, \dots, v_{M_-} are thus distributed identically to the weights of the negative examples in the scenario of Theorem 18 conditioned on $m_- = M_-$, and likewise for the u_1, \dots, u_{M_+} and the positive examples.

We define the following event:

- Event A_{M_-, M_+} : $u_1 \geq v_1 + B$.

For succinctness let us write A_m for the event (in our original scenario of a size- m sample S drawn from \mathcal{U}) that S is positive-skewed. We then have

$$\begin{aligned} \Pr[A_m] &\geq \Pr[.49m < m_-, m_+ < .51m] \cdot \Pr[A_m \mid .49m < m_-, m_+ < .51m] \\ &\geq (1 - 2^{-\Omega(m)}) \Pr[A_m \mid .49m < m_-, m_+ < .51m] \\ &\geq (1 - 2^{-\Omega(m)}) \min_{.49m < M_-, M_+ < .51m} \Pr[A_m \mid m_- = M_- \text{ and } m_+ = M_+] \\ &= (1 - 2^{-\Omega(m)}) \min_{.49m < M_-, M_+ < .51m} \Pr[A_{M_-, M_+}]. \end{aligned}$$

where the second inequality holds by Chernoff bound.

It thus suffices to show that for any values M_-, M_+ in $(.49m, .51m)$ we have $\Pr[A_{M_-, M_+}] \geq 0.0291$. Fix any M_-, M_+ in this range; we will henceforth only consider the experiment E_{M_-, M_+} in which any event involving only the u_i 's is independent from any event involving only v_i 's.

Let n' denote $n-1$. The idea of the next part of the proof is to show that with some probability v_1 falls into a relatively small left tail of the distribution while u_1 is bounded away from this tail. This gives us a gap between u_1 and v_1 as desired.

We consider u_1 first. For $1 \leq i \leq n'$ let $\psi(i)$ denote $\sum_{j=0}^{i-1} \binom{n'}{j} 2^{-n'}$. Note that $\psi(i)$ is precisely the weight in the ‘‘left tail up to i ’’ of the distribution $1 + B(n', \frac{1}{2})$. Let X be the event that $\psi(u_1) \geq \frac{1}{2m}$

3. Note that this is not the case in S because the total number of examples is m so that more positive examples means less negative examples and vice versa. This dependence affects that probability over weights for the lightest positive and negative examples in a subtle way which is hard to analyze directly.

and $u_1 \leq n'/2$. In order to have $\psi(u_1) < \frac{1}{2m}$, at least one of the $M_+ < .51m$ draws from $1 + B(n', \frac{1}{2})$ must land in the “left tail” of weight less than $\frac{1}{2m}$; by a union bound the probability that this occurs is less than $\frac{0.51}{2}$ and hence $\Pr[\psi(u_1) \geq \frac{1}{2m}] \geq 1 - \frac{0.51}{2} > 0.745$. The probability that $u_1 \geq n'/2$ is $2^{-\Omega(m)}$ and thus $\Pr[X] > 0.745 - 2^{-\Omega(m)} > 0.74$.

Next consider v_1 . For $1 \leq i \leq n'$ let $\phi(i)$ denote $\sum_{j=0}^i \binom{n'}{j} 2^{-n'}$; similar to $\psi(i)$ we have that $\phi(i)$ captures the weight in the left tail of $B(n', \frac{1}{2})$. Let Y be the event that $\phi_n(v_1) \leq \frac{1}{4m}$. This event fails to occur only if each of the M_- draws from $B(n', \frac{1}{2})$ misses the left tail of weight at most $\frac{1}{4m}$. We need to be slightly careful; note that $\phi(\cdot)$ takes discrete values, so this tail may actually weigh less than $\frac{1}{4m}$ (e.g. conceivably $\phi(22) = \frac{1}{m^2}$ and $\phi(23) = \frac{1}{m}$.) To take care of this we will now show that this tail cannot weigh much less than $\frac{1}{4m}$.

For $c \geq 1$ let $\sigma(c)$ denote the largest integer such that $\phi(\sigma(c)) \leq \frac{1}{cm}$.

Lemma 19 *For any constant $c \geq 1$ we have $\phi(\sigma(c)) \geq \frac{1}{3cm}$.*

Proof Suppose not; then we have $\phi(\sigma(c)) < \frac{1}{3cm}$ and $\phi(\sigma(c) + 1) > \frac{1}{cm}$. This implies that $\binom{n'}{\sigma(c)+1} > 2 \sum_{j=0}^{\sigma(c)} \binom{n'}{j}$ so in particular $\binom{n'}{\sigma(c)+1} > 2 \binom{n'}{\sigma(c)}$. This implies that $n' - \sigma(c) > 2\sigma(c) + 2$ which implies $\sigma(c) < (n' - 2)/3$. But then Chernoff bound implies that for such values of $\sigma(c)$, $\phi(\sigma(c) + 1) = 2^{-\Omega(n')}$ which contradicts the inequality $\phi(\sigma(c) + 1) > \frac{1}{cm}$ since c is constant and m is polynomial in n . \blacksquare

The lemma implies that the left tail of weight at most $\frac{1}{4m}$ must have weight at least $\frac{1}{12m}$. Hence the probability that each of the $M_- > .49m$ draws from $B(n', \frac{1}{2})$ misses this left tail is at most $(1 - \frac{1}{12m})^{.49m}$. This is at most 0.96 and hence $\Pr[Y] \geq 0.04$.

We next show that if events X and Y both occur then event A_{M_-, M_+} occurs. This will complete the proof of the theorem since the events X and Y are independent and we have that $\Pr[A_{M_-, M_+}] \geq \Pr[X] \Pr[Y] \geq 0.0296$.

Suppose, for the sake of contradiction, that events X and Y both occur but $u_1 \leq v_1 + (B - 1)$. Since X occurs we have $\psi(u_1) \geq \frac{1}{2m}$, i.e.

$$\Psi(u_1) = \sum_{j=0}^{u_1-1} \binom{n'}{j} 2^{-n'} \geq \frac{1}{2m}.$$

On the other hand since Y occurs we have $\phi(v_1) \leq \frac{1}{4m}$, so

$$\sum_{j=0}^{v_1} \binom{n'}{j} 2^{-n'} \leq \frac{1}{4m}. \quad (5)$$

These two inequalities together clearly imply $u_1 > v_1$. In fact they imply

$$\sum_{j=v_1+1}^{u_1-1} \binom{n'}{j} 2^{-n'} \geq \frac{1}{4m}. \quad (6)$$

Thus we see that the weights between $v_1 + 1$ and $u_1 - 1$ have a substantial size. We next show that this implies that the weights below v_1 also have a substantial size, contradicting Equation (5). The following lemma is useful:

Lemma 20 For all j such that $u_1 - 3B \leq j \leq u_1 - 1$ we have $\binom{n'}{j} \geq \frac{1}{2} \binom{n'}{u_1-1}$.

Proof Clearly it suffices to prove that $2 \binom{n'}{u_1-3B} \geq \binom{n'}{u_1-1}$. By event X we know that $\psi(u_1) \geq \frac{1}{2m}$. But the left tail Chernoff bound implies that unless

$$u_1 - 1 \geq \frac{n'}{2} - 2\sqrt{n' \log m} \quad (7)$$

we have $\psi(u_1) < \frac{1}{m^4} < \frac{1}{2m}$ so (7) must hold.

Let $c = \frac{n'}{2} - (u_1 - 1)$ so $0 < c \leq 2\sqrt{n' \log m}$. Now observe that for any b such that $b < 0.1n'$ we have

$$\frac{\binom{n'}{n'/2-b}}{\binom{n'}{n'/2-b-1}} = \frac{n'/2+b+1}{n'/2-b} = 1 + \frac{2b+1}{n'/2-b} < 1 + \frac{2b+1}{0.4n'} = 1 + \frac{5b+2.5}{n'}.$$

We thus have

$$\begin{aligned} \frac{\binom{n'}{u_1-1}}{\binom{n'}{u_1-3B}} &= \frac{\binom{n'}{u_1-1}}{\binom{n'}{u_1-2}} \cdot \frac{\binom{n'}{u_1-2}}{\binom{n'}{u_1-3}} \cdots \frac{\binom{n'}{u_1-3B+1}}{\binom{n'}{u_1-3B}} \\ &= \frac{\binom{n'}{\frac{n'}{2}-c}}{\binom{n'}{\frac{n'}{2}-c-1}} \cdot \frac{\binom{n'}{\frac{n'}{2}-c-1}}{\binom{n'}{\frac{n'}{2}-c-2}} \cdots \frac{\binom{n'}{\frac{n'}{2}-(c+3B-2)}}{\binom{n'}{\frac{n'}{2}-(c+3B-3)}} \\ &< \left(1 + \frac{5c+2.5}{n'}\right) \left(1 + \frac{5(c+1)+2.5}{n'}\right) \cdots \left(1 + \frac{5(c+3B-2)+2.5}{n'}\right) \\ &< \left(1 + \frac{5(c+3B)+2.5}{n'}\right)^{3B} \\ &\leq e^{\frac{5(c+3B)+2.5}{n'} 3B} \end{aligned}$$

where we have used the inequality $1+x \leq e^x$. The last quantity is at most $\sqrt{e} < 2$ provided that

$$3B < \frac{n'}{10(c+3B)+5}. \quad (8)$$

Now since $c \leq 2\sqrt{n' \log m}$ and we can bound $5 < \sqrt{n' \log m}$ and $30B < 0.5\sqrt{n' \log m}$ this holds if

$$3B = \frac{1}{22} \sqrt{\frac{n}{\log m}} < \frac{n'}{21.5\sqrt{n' \log m}} = \frac{1}{21.5} \sqrt{\frac{n'}{\log m}}$$

which is clearly true for sufficiently large n . ■

Recalling that $u_1 \leq v_1 + (B-1)$, we have that the sum in Equation (6) has at most $B-2$ terms. Now since event X holds, $u_1 < \frac{n'}{2}$ and therefore the largest of these terms is $\binom{n'}{u_1-1} 2^{-n'}$. By Equation (6) we thus have that

$$\binom{n'}{u_1-1} 2^{-n'} \geq \frac{1}{4(B-2)m}. \quad (9)$$

Now Lemma 20 together with Equation (9), implies that we have

$$\sum_{j=u_1-3B}^{v_1-1} \binom{n'}{j} 2^{-n'} > \sum_{j=u_1-3B}^{u_1-B-1} \binom{n'}{j} 2^{-n'} \geq \sum_{j=u_1-3B}^{u_1-B-1} \frac{1}{2} \binom{n'}{u_1-1} 2^{-n'} \geq \frac{2B}{2} \frac{1}{4(B-2)m} > \frac{1}{4m}$$

but this contradicts Equation (5). ■

4.2 Lower Bound for Large k

Using the fact that the sample is positive-skewed with constant probability we can prove the lower bound along the same lines as before.

Definition 21 A sample S is a \mathcal{U} -typical sample if

- Every example $x \in S$ satisfies $0.49n \leq |x| \leq 0.51n$.
- Every pair of examples $x^{i,+}$ and $x^{j,-}$ in S satisfies $x^{i,+} \cdot x^{j,-} \leq 0.26n$.

As above we can apply Chernoff bounds to derive the next two lemmas:

Lemma 22 For $m = \text{poly}(n)$, with probability $1 - 2^{-\Omega(n)}$ a random i.i.d. sample of m draws from \mathcal{U} is a \mathcal{U} -typical sample.

Definition 23 Let S be a sample. The set $Z(S)$ includes all positive examples z such that every example x in S satisfies $x \cdot z \leq 0.26n$.

Lemma 24 Let S be a \mathcal{U} -typical sample of size $m = \text{poly}(n)$ examples. Then $\Pr_{\mathcal{U}}[z \in Z(S) | f(z) = 1] = 1 - 2^{-\Omega(n)}$.

The following lemma is analogous to Lemma 11:

Lemma 25 Let S be a \mathcal{U} -typical sample of size m . Then the maximum margin m_S satisfies

$$m_S \geq \frac{1}{2} \left(\frac{1}{\sqrt{m}} \sqrt{\rho_k(u_1)} - \sqrt{m \rho_k(.26n)} \right).$$

Proof We exhibit an explicit linear threshold function h' which has this margin. Let $h'(x) = \text{sign}(W' \cdot \phi_k(x) - \theta')$ be defined as follows:

- For each positive example $x^{i,+}$ in S , pick a set of $\rho_k(u_1)$ features (monomials) which take value 1 on $x^{i,+}$. This can be done since each positive example $x^{i,+}$ has at least u_1 bits which are 1. For each feature T in each of these sets, assign $W'_T = 1$.
- For all remaining features T set $W'_T = 0$.
- Set θ' to be the value that gives the maximum margin on $\phi_k(S)$ for this W' , i.e. θ' is the average of the smallest value of $W' \cdot \phi_k(x^{i,+})$ and the largest value of $W' \cdot \phi_k(x^{j,-})$.

Note that since each positive example contributes at most $\rho_k(u_1)$ nonzero coefficients to W' , the number of 1's in W' is at most $m\rho_k(u_1)$, and hence $\|W'\| \leq \sqrt{m\rho_k(u_1)}$. By construction we also have that each positive example $x^{i,+}$ satisfies $W' \cdot \phi_k(x^{i,+}) \geq \rho_k(u_1)$.

Since S is a \mathcal{U} -typical sample, each negative example $x^{j,-}$ in S shares at most $.26n$ ones with any positive example in S . Hence the value of $W' \cdot \phi_k(x^{j,-})$ is a sum of at most $m\rho_k(.26n)$ numbers whose squares sum to at most $m\rho_k(u_1)$. By Observation 1 we have that $W' \cdot \phi_k(x^{j,-}) \leq \sqrt{m\rho_k(.26n)}\sqrt{m\rho_k(u_1)}$.

The lemma follows by combining the above bounds on $\|W'\|$, $W' \cdot \phi_k(x^{i,+})$ and $W' \cdot \phi_k(x^{j,-})$. ■

Now we can give a lower bound on the threshold θ for the maximum margin classifier.

Lemma 26 *Let S be a labeled sample of size m which is \mathcal{U} -typical and positive skewed, and let $h(x) = \text{sign}(W \cdot \phi_k(x) - \theta)$ be the maximum margin hypothesis for S . Then*

$$\theta \geq \frac{1}{2} \left(\frac{1}{\sqrt{m}} \sqrt{\rho_k(u_1)} - \sqrt{m\rho_k(.26n)} \right) - \sqrt{\rho_k(u_1 - B)}.$$

Proof Since S is positive-skewed we know that $W \cdot \phi_k(x^{1,-})$ is a sum of at most $\rho_k(u_1 - B)$ weights W_T , and since W is normalized the sum of the squares of these weights is at most 1. By Observation 1 we thus have $W \cdot \phi_k(x^{1,-}) \geq -\sqrt{\rho_k(u_1 - B)}$. Now since $\theta \geq W \cdot \phi_k(x^{1,-}) + m_S$, Lemma 25 implies the result. ■

Putting all of the pieces together, we have:

Theorem 27 *If the maximum margin algorithm uses the kernel K_k for $k = \omega(\sqrt{n} \log^{\frac{3}{2}} n)$ when learning $f(x) = x_1$ under the uniform distribution then with probability at least 0.028 its hypothesis has error $\epsilon = \frac{1}{2} - 2^{-\Omega(n)}$.*

Proof By Lemma 22 and Theorem 18, the sample S used for learning is both \mathcal{U} -typical and positive skewed with probability at least $0.029 - 1/2^{-\Omega(n)}$ which is more than 0.028 for sufficiently large n . Consider any $z \in Z(S)$. Using the reasoning from Lemma 13, $W \cdot \phi(z)$ is a sum of at most $m\rho_k(.26n)$ numbers whose squares sum to at most 1, so $W \cdot \phi(z) \leq \sqrt{m\rho_k(.26n)}$. The example z is erroneously classified as negative by h if

$$\frac{1}{2} \left(\frac{1}{\sqrt{m}} \sqrt{\rho_k(u_1)} - \sqrt{m\rho_k(.26n)} \right) - \sqrt{\rho_k(u_1 - B)} > \sqrt{m\rho_k(.26n)}.$$

so it suffices to show that

$$\sqrt{\rho_k(u_1)} > 3m \left(\sqrt{\rho_k(.26n)} + \sqrt{\rho_k(u_1 - B)} \right). \quad (10)$$

Recall that $\rho_k(x) = \sum_{j=0}^k \binom{x}{j}$. Note that for $k = n$ (all-monomials kernel) the above inequality becomes $2^{u_1/2} > 3m (2^{.13n} + 2^{(u_1-B)/2})$ which is clearly true. In Appendix B we show that Equation (10) holds for all $k = \omega(\sqrt{n} \log^{\frac{3}{2}} n)$ as required.

The above argument shows that any $z \in Z(S)$ is misclassified, and Lemma 24 guarantees that the relative weight of $Z(S)$ in positive examples is $1 - 2^{-\Omega(n)}$. Since $\Pr_{x \in \mathcal{U}}[f(x) = 1]$ is $1/2$, we have

that with probability at least 0.028 the hypothesis h has error rate at least $\epsilon = \frac{1}{2} - 2^{-\Omega(n)}$, and we are done. ■

Remark 28 Here again we can adapt the proofs to show non-learnability results for the polynomial kernel $K_k(x, y) = (x \cdot y)^k$. We modify the definition of W' in Lemma 25 as follows. For every positive example $x^{i,+}$ in the sample let $\hat{x}^{i,+}$ be the example obtained by picking an arbitrary subset of size u_1 of the original true bits and setting all other bits to 0. Now let $W' = \sum_{x^{i,+} \in S} \phi(\hat{x}^{i,+})$. Arguing as in Remark 16 we get that the maximum margin is at least

$$\frac{1}{2} \cdot \frac{u_1^k - m(0.26n)^k}{m\sqrt{u_1^k}}.$$

Now in Lemma 26 we get that $W' \cdot \phi(x^{1,-}) \geq -(u_1 - B)^{k/2}$ which again implies a lower bound on the threshold.

Finally, following Theorem 27 and the argument in Remark 16 one can show that for an example $z \in Z(S)$ we have $W \cdot \phi(z) \leq m\sqrt{(0.26n)^k}$ so that z is misclassified if

$$u_1^k - m(0.26n)^k - 2m\sqrt{u_1^k}\sqrt{(u_1 - B)^k} \geq 2m^2\sqrt{u_1^k(0.26n)^k}$$

which is true if

$$u_1^{k/2} > 5m^2(u_1 - B)^{k/2}.$$

Using the reasoning in Case 1 of Appendix B, one can show that this holds for $k = \omega(\sqrt{n} \log^{\frac{3}{2}} n)$.

5. Conclusions and Future Work

Boolean kernels offer an interesting new algorithmic approach to one of the major open problems in computational learning theory, namely learnability of DNF expressions. We have studied the performance of the maximum margin algorithm with the Boolean kernels, giving negative results for several settings of the problem. Our results indicate that the maximum margin algorithm can overfit even when learning simple target functions and using natural and expressive kernels for such functions, and even when combined with structural risk minimization. Our results consider cases where the L_2 norm of examples in the expanded feature space is large. This seems necessary for learning DNF; note that while one can use an exponential function to define a kernel with weighted monomials where the weight decays exponentially depending on the degree k , this implies that the margin for functions of high degree is exponentially small.

While our results are negative there are several interesting avenues suggested by this work which may succeed; we discuss these briefly below. One direction is to modify the basic learning algorithm. Many interesting variants of the basic maximum margin algorithm have been used in recent years, such as soft margin criteria and kernel regularization. It may be possible to prove positive results for some DNF learning problems using these approaches. A starting point would be to test their performance on the counterexamples (functions and distributions) which we have constructed.

A more immediate goal is to close the gap between small and large k in our results for the uniform distribution. It is well known (see, e.g., Verbeurgt, 1990) that when learning polynomial

size DNF under the uniform distribution, conjunctions of length $\omega(\log n)$ can be ignored with little effect. Hence the most interesting setting of k for the uniform distribution learning problem is $k = \Theta(\log n)$. Learning under the uniform distribution with a $k = \Theta(\log n)$ kernel is qualitatively quite different from learning with the large values of k which we were able to analyze. For example, for $k = \Theta(\log n)$ if a sufficiently large polynomial size sample is taken, then with very high probability all features (monomials of size at most k) are active in the sample.

As a first concrete problem in this scenario, one might consider the question of whether a $k = \Theta(\log n)$ kernel maximum margin algorithm can efficiently PAC learn the target function $f(x) = x_1$. For this problem it is easy to show that the naive hypothesis h' constructed in our proofs achieves both a large margin and high accuracy. Moreover, it is possible to show that with high probability the maximum margin hypothesis has a margin which is within a multiplicative factor of $(1 + o(1))$ of the margin achieved by h' . Though these preliminary results do not answer the above question they suggest that the answer may be positive. A positive answer, in our view, would be strong motivation to analyze the general case.

Finally, the kernel we have used is natural in terms of capturing all monomials of a certain length but there are other ways to capture natural kernels for Boolean problems. An interesting possibility is using a kernel of parity functions and such a construction can indeed be given. The resulting representation is closely related to learning via the Fourier transform as done in the work of Linial et al. (1993); Kushilevitz and Mansour (1993); Mansour (1995) but the algorithmic ideas are very different to the ones used by maximum margin algorithms.

Acknowledgments

R.K.'s work was partly supported by NSF Grant IIS-0099446, and by a Research Semester Fellowship Award from Tufts University. Much of R.A.S.'s work was done while supported by a National Science Foundation Mathematical Sciences Postdoctoral Research Fellowship at Harvard University.

Appendix A. Proof of Equation (3)

To show that

$$\rho_k(.99n^{2/3}) > 2m\sqrt{\rho_k(1.01n^{1/3})\rho_k(1.01n^{2/3})} + m\rho_k(1.01n^{1/3})$$

it suffices to show that

$$\rho_k(.99n^{2/3}) > 3m\sqrt{\rho_k(1.01n^{1/3})\rho_k(1.01n^{2/3})}. \tag{11}$$

The proof uses several cases depending on the value of k relative to n .

Case 1: $k \leq 0.505n^{1/3}$. Since $\rho_k(\ell) = \sum_{i=1}^k \binom{\ell}{i}$, for $k \leq \ell/2$ we have that $\rho_k(\ell) \leq k\binom{\ell}{k}$. For all k we have $\rho_k(\ell) \geq \binom{\ell}{k}$ so it suffices to show that

$$\binom{.99n^{2/3}}{k} > 3mk\sqrt{\binom{1.01n^{1/3}}{k}\binom{1.01n^{2/3}}{k}}$$

which is equivalent (clearing denominators from the binomial coefficients) to

$$\prod_{i=0}^{k-1} (.99n^{2/3} - i) > 3mk \sqrt{\prod_{i=0}^{k-1} (1.01n^{1/3} - i)(1.01n^{2/3} - i)}.$$

We now use the fact that for $i \geq 0$ we have $(A - i)(B - i) \leq (\sqrt{AB} - i)^2$ provided that $2\sqrt{AB} < A + B$; it is easy to see that this latter condition holds for $A = 1.01n^{1/3}$, $B = 1.01n^{2/3}$. It thus suffices to show that

$$\prod_{i=0}^{k-1} (.99n^{2/3} - i) > 3mk \prod_{i=0}^{k-1} (1.01n^{1/2} - i)$$

which in turn is implied by

$$\left(\frac{.99n^{2/3}}{1.01n^{1/2}} \right)^k > 3mn$$

(we used the fact that $k \leq n$ to obtain the right-hand side above). This holds as long as $k > \frac{\log(3mn)}{\log 0.98 + \frac{1}{6} \log n} = \Theta(1)$ for any $m = \text{poly}(n)$. Therefore the condition holds for any $k = \omega(1)$.

Case 2: $0.5 \cdot 1.01n^{1/3} \leq k \leq 5 \cdot 1.01n^{1/3}$. In this case we use the bounds $\binom{\ell}{k}^k \leq \rho_k(\ell) = \sum_{i=1}^k \binom{\ell}{i} \leq \left(\frac{e\ell}{k}\right)^k$ for the first and third occurrences of ρ_k in equation (11) and we use $\rho_k(\ell) \leq 2^\ell$ for the second occurrence. It thus suffices to show that

$$\left(\frac{.99n^{2/3}}{k} \right)^k > 3m \sqrt{\left(\frac{e \cdot 1.01n^{2/3}}{k} \right)^k} \cdot 2^{1.01n^{1/3}}.$$

Applying the upper bound on k in the denominator on the left side, and the lower bound on k in the denominator on the right side, it suffices to show that

$$\left(\frac{.99}{5.05} n^{1/3} \right)^k > 3m \sqrt{\left(\frac{e \cdot 1.01}{0.505} n^{1/3} \right)^k} \cdot 2^{1.01n^{1/3}}$$

Now since $1.01n^{1/3} \leq 2k$ the condition holds if

$$\left(\frac{n^{1/3}}{6} \right)^k > 3m \left(2e \cdot n^{1/3} \right)^{k/2} \cdot 2^k$$

or equivalently if

$$\left(\frac{n^{1/6}}{12\sqrt{2e}} \right)^k > 3m.$$

This obviously holds since $k = \Theta(n^{1/3})$.

Case 3: $5 \cdot 1.01n^{1/3} \leq k \leq 0.25 \cdot 0.99n^{2/3}$. We use the same bounds as in the previous case to start the analysis, so we want to show that

$$\left(\frac{.99n^{2/3}}{k} \right)^k > 3m \sqrt{\left(\frac{e \cdot 1.01n^{2/3}}{k} \right)^k} \cdot 2^{1.01n^{1/3}}.$$

Since $1.01n^{1/3} \leq k/5$ it suffices to show that

$$\left(\frac{.99n^{2/3}}{k}\right)^k > 3m \left(\frac{e \cdot 1.01n^{2/3}}{k}\right)^{k/2} \cdot 2^{k/10}$$

which holds (taking k -th roots and rearranging) if and only if

$$\left(\frac{1}{2}\right)^{1/10} \cdot \frac{.99n^{2/3}}{k} \cdot \frac{\sqrt{k}}{n^{1/3}\sqrt{1.01 \cdot e}} = \left(\frac{1}{2}\right)^{1/10} \cdot \left(\frac{.99}{\sqrt{1.01 \cdot e}}\right) \cdot \frac{n^{1/3}}{\sqrt{k}} > (3m)^{1/k}.$$

Using our upper bound on k on the left side, the previous inequality holds if

$$\left(\frac{1}{2}\right)^{1/10} \frac{.99}{\sqrt{1.01 \cdot e}} \cdot \frac{2}{\sqrt{.99}} > (3m)^{1/k}$$

and since the left side is greater than 1.1 the inequality holds if $k > \frac{\log 3m}{\log 1.1} = \Theta(\log n)$ for $m = \text{poly}(n)$. This obviously holds since $k = \Omega(n^{1/3})$.

Case 4: $0.25 \cdot 0.99n^{2/3} \leq k \leq 0.5 \cdot 0.99n^{2/3}$. We use the following bound (proved later) which holds for $0 < \alpha < 1$:

$$\sum_{i=1}^{\alpha q} \binom{q}{i} \geq \frac{1}{\sqrt{2\pi q}} 2^{H(\alpha)q} \quad (12)$$

where $H(p) = -p \log p - (1-p) \log(1-p)$ is the binary entropy function. Applying this bound to the left side of (11) with $q = .99n^{2/3}$ and $\alpha = k/q$, we have $.25 \leq \alpha \leq .5$ so $H(\alpha) > .81$. Since $\rho_k(\ell)$ is always at most 2^ℓ it suffices to show that

$$\frac{1}{\sqrt{2\pi \cdot .99n^{2/3}}} 2^{0.81 \cdot 0.99n^{2/3}} > 3m \sqrt{2^{1.01n^{2/3} + 1.01n^{1/3}}}.$$

This is easily seen to hold for any $m = \text{poly}(n)$.

To prove the bound (12) we use Stirling's approximation $\sqrt{2\pi n} \left(\frac{n}{e}\right)^n \leq n! \leq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \sqrt{1 + \frac{1}{2n}}$; in fact we use a weaker form with $\sqrt{2}$ instead of $\sqrt{1 + \frac{1}{2n}}$ in the upper bound. We thus have

$$\begin{aligned} \sum_{i=1}^{\alpha q} \binom{q}{i} &\geq \binom{q}{\alpha q} = \frac{q!}{(\alpha q)!((1-\alpha)q)!} \geq \frac{\sqrt{2\pi q}}{2\sqrt{2\pi\alpha q}\sqrt{2\pi(1-\alpha)q}} \left(\frac{q}{e}\right)^q \left(\frac{e}{\alpha q}\right)^{\alpha q} \left(\frac{e}{(1-\alpha)q}\right)^{(1-\alpha)q} \\ &= \frac{1}{2\sqrt{2\pi\alpha(1-\alpha)q}} \alpha^{-\alpha q} (1-\alpha)^{-(1-\alpha)q} = \frac{1}{2\sqrt{2\pi\alpha(1-\alpha)q}} 2^{qH(\alpha)}. \end{aligned}$$

Equation (12) follows since $\alpha(1-\alpha) \leq 1/4$.

Note that by using $\sum_{i=0}^{\alpha q} \binom{q}{i} \leq \alpha q \binom{q}{\alpha q}$ one can also obtain $\sum_{i=0}^{\alpha q} \binom{q}{i} \leq \frac{\sqrt{\alpha q}}{\sqrt{\pi(1-\alpha)}} 2^{H(\alpha)q}$.

Case 5: $k \geq 0.5 \cdot 0.99n^{2/3}$. In this case we have $\rho_k(.99n^{2/3}) = \sum_{i=1}^k \binom{.99n^{2/3}}{i} \geq \frac{1}{2} 2^{.99n^{2/3}}$. Thus it suffices to show that

$$\frac{1}{2} \cdot 2^{.99n^{2/3}} > 3m \sqrt{2^{1.01n^{2/3} + 1.01n^{1/3}}}$$

which is easily seen to hold for any $m = \text{poly}(n)$. Thus Equation (11) holds for all $k = \omega(1)$. \blacksquare

Appendix B. Proof of Equation (10)

We must show that $\sqrt{\rho_k(u_1)} > 3m \left(\sqrt{\rho_k(.26n)} + \sqrt{\rho_k(u_1 - B)} \right)$. Since we are assuming that the sample S is \mathcal{U} -typical, we have $u_1 \geq .49n$ so $u_1 - B > 0.26n$. It thus suffices to show that $\rho_k(u_1) > 36m^2 \rho_k(u_1 - B)$.

Case 1: $k \leq \frac{1}{2}(u_1 - B)$. Since $\rho_k(\ell) = \sum_{i=1}^k \binom{\ell}{i}$, for $k \leq \ell/2$ we have $\rho_k(\ell) \leq k \binom{\ell}{k}$. Also for all k , $\rho_k(\ell) \geq \binom{\ell}{k}$ so it suffices to show that

$$\binom{u_1}{k} > 36m^2 k \binom{u_1 - B}{k}.$$

This inequality is true if

$$\left(\frac{u_1}{u_1 - B} \right)^k > 36m^2 k.$$

Recall that $B = \frac{1}{66} \sqrt{\frac{n}{\log m}}$. Now using the fact that

$$\frac{u_1}{u_1 - B} = 1 + \frac{B}{u_1 - B} > 1 + \frac{B}{n} = 1 + \frac{1}{66\sqrt{n \log m}}$$

it suffices to show that

$$\left(1 + \frac{1}{66\sqrt{n \log m}} \right)^k > 36m^2 k.$$

Using the fact that $1 + x \geq e^{x/2}$ for $0 < x < 1$, we can see that this inequality holds if

$$k > 132 \sqrt{n \log(m)} \ln(36m^2 n).$$

Since $m = \text{poly}(n)$, this is the case for $k = \omega(\sqrt{n} \log^{\frac{3}{2}} n)$.

Case 2: $\frac{1}{2}(u_1 - B) < k$. Since $\rho_k(u_1 - B) \leq 2^{u_1 - B}$, it suffices to show that

$$\sum_{i=1}^{\frac{u_1 - B}{2}} \binom{u_1}{i} > 36m^2 \cdot 2^{u_1 - B}.$$

Since $\sqrt{u_1} > \sqrt{0.49n} > \frac{92}{132} \sqrt{n} > 92 \frac{B}{2}$ it suffices to show that

$$\sum_{i=1}^{\frac{u_1 - \sqrt{u_1}}{92}} \binom{u_1}{i} > 36m^2 \cdot 2^{u_1 - B}.$$

Using Stirling approximation it is easy to check that $\binom{q}{q/2} < \sqrt{1 + \frac{1}{2q}} \sqrt{\frac{2}{\pi}} \frac{1}{\sqrt{q}} 2^q$ and this implies that

$$\sum_{i=1}^{\frac{u_1 - \sqrt{u_1}}{92}} \binom{u_1}{i} > \frac{1}{2} 2^{u_1} - \frac{\sqrt{u_1}}{92} \sqrt{1 + \frac{1}{2u_1}} \sqrt{\frac{2}{\pi}} \frac{1}{\sqrt{u_1}} 2^{u_1} > 0.49 \cdot 2^{u_1}$$

so the condition above holds if

$$0.49 \cdot 2^B > 36m^2.$$

This is clearly true since $m = \text{poly}(n)$ and $B = \frac{1}{66} \sqrt{\frac{n}{\log m}}$. ■

References

- S. Ben-David, N. Eiron, and H.-U. Simon. Limitations of learning via embeddings in euclidean half spaces. *Journal of Machine Learning Research*, 3:441–461, 2002.
- A. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM Journal on Computing*, 23(5):990–1000, 1994.
- A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the Twenty-Sixth Annual Symposium on Theory of Computing*, pages 253–262, 1994.
- A. Blum and S. Rudich. Fast learning of k -term DNF formulas with queries. *Journal of Computer and System Sciences*, 51(3):367–373, 1995.
- A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam’s razor. *Information Processing Letters*, 24:377–380, 1987.
- B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.
- N. Bshouty. A subexponential exact learning algorithm for DNF using equivalence queries. *Information Processing Letters*, 59:37–39, 1996.
- N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.
- J. Forster, N. Schmitt, H.-U. Simon, and T. Suttrop. Estimating the optimal margins of embeddings in euclidean half spaces. *Machine Learning*, 51(3):263–281, 2003.
- T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. In *Proceedings of the 15th International Conference on Machine Learning*, pages 188–196, 1998.
- C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.
- T. Hancock and Y. Mansour. Learning monotone k - μ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Conference on Computational Learning Theory*, pages 179–193, 1991.
- J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55:414–440, 1997.
- M. Kearns and U. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, 1994.
- R. Khardon. On using the Fourier transform to learn disjoint DNF. *Information Processing Letters*, 49:219–222, 1994.

- R. Khardon, D. Roth, and R. Servedio. Efficiency versus convergence of Boolean kernels for on-line learning algorithms. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
- A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. In *Proceedings of the Thirty-Third Annual Symposium on Theory of Computing*, pages 258–265, 2001.
- A. Kowalczyk, A. J. Smola, and R. C. Williamson. Kernel machines and Boolean functions. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
- L. Kucera, A. Marchetti-Spaccamela, and M. Protassi. On learning monotone DNF formulae under uniform distributions. *Information and Computation*, 110:84–95, 1994.
- E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. on Computing*, 22(6):1331–1348, 1993.
- E. Kushilevitz and D. Roth. On learning visual concepts and DNF formulae. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 317–326, 1993.
- N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.
- Y. Mansour. An $o(n^{\log \log n})$ learning algorithm for DNF under the uniform distribution. *Journal of Computer and System Sciences*, 50:543–550, 1995.
- M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry*. MIT Press, Cambridge, MA, 1968.
- K. Sadohara. Learning of Boolean functions using support vector machines. In *Proc. of the 12th International Conference on Algorithmic Learning Theory*, pages 106–118. Springer, 2001. LNAI 2225.
- Y. Sakai and A. Maruoka. Learning monotone log-term DNF formulas under the uniform distribution. *Theory of Computing Systems*, 33:17–33, 2000.
- R. Servedio. On PAC learning using Winnow, Perceptron, and a Perceptron-like algorithm. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 296–307, 1999.
- R. Servedio. On learning monotone DNF under product distributions. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 473–489, 2001.
- J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.
- J. Shawe-Taylor and N. Cristianini. *An introduction to support vector machines*. Cambridge University Press, 2000.
- J. Tarui and T. Tsukiji. Learning DNF by approximating inclusion-exclusion formulae. In *Proceedings of the Fourteenth Conference on Computational Complexity*, pages 215–220, 1999.

- L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326, 1990.
- K. Verbeurgt. Learning sub-classes of monotone DNF on the uniform distribution. In *Proceedings of the Ninth Conference on Algorithmic Learning Theory*, pages 385–399, 1998.
- C. Watkins. Kernels from matching operations. Technical Report CSD-TR-98-07, Computer Science Department, Royal Holloway, University of London, 1999.

A Bayes Optimal Approach for Partitioning the Values of Categorical Attributes

Marc Boullé

*France Telecom R&D
2 Avenue Pierre Marzin
22300 Lannion, France*

MARC.BOULLE@FRANCETELECOM.COM

Editor: Greg Ridgeway

Abstract

In supervised machine learning, the partitioning of the values (also called grouping) of a categorical attribute aims at constructing a new synthetic attribute which keeps the information of the initial attribute and reduces the number of its values. In this paper, we propose a new grouping method MODL¹ founded on a Bayesian approach. The method relies on a model space of grouping models and on a prior distribution defined on this model space. This results in an evaluation criterion of grouping, which is minimal for the most probable grouping given the data, *i.e.* the Bayes optimal grouping. We propose new super-linear optimization heuristics that yields near-optimal groupings. Extensive comparative experiments demonstrate that the MODL grouping method builds high quality groupings in terms of predictive quality, robustness and small number of groups.

Keywords: data preparation, grouping, Bayesianism, model selection, classification, naïve Bayes

1 Introduction

Supervised learning consists of predicting the value of a class attribute from a set of explanatory attributes. Many induction algorithms rely on discrete attributes and need to discretize continuous attributes or to group the values of categorical attributes when they are too numerous. While the discretization problem has been studied extensively in the past, the grouping problem has not been explored so deeply in the literature. The grouping problem consists in partitioning the set of values of a categorical attribute into a finite number of groups. For example, most decision trees exploit a grouping method to handle categorical attributes, in order to increase the number of instances in each node of the tree (Zighed and Rakotomalala, 2000). Neural nets are based on continuous attributes and often use a 1-to-N binary encoding to preprocess categorical attributes. When the categories are too numerous, this encoding scheme might be replaced by a grouping method. This problem arises in many other classification algorithms, such as Bayesian networks or logistic regression. Moreover, the grouping is a general-purpose method that is intrinsically useful in the data preparation step of the data mining process (Pyle, 1999).

The grouping methods can be clustered according to the search strategy of the best partition and to the grouping criterion used to evaluate the partitions. The simplest algorithm tries to find the best bipartition with one category against all the others. A more interesting approach consists in searching a bipartition of all categories. The Sequential Forward Selection method derived from that of Cestnik et al. (1987) and evaluated by Berckman (1995) is a greedy algorithm that initializes a group with the best category (against the others), and iteratively adds new categories to this first group. When the class attribute has two values, Breiman et al. (1984) have proposed in CART an optimal method to group the categories into two groups for the Gini criterion. This

¹ This work is covered under French patent number 04 00179.

algorithm first sorts the categories according to the probability of the first class value, and then searches for the best split in this sorted list. This algorithm has a time complexity of $O(I \log(I))$, where I is the number of categories. Based on the ideas presented in (Lechevallier, 1990; Fulton et al., 1995), this result can possibly be extended to find the optimal partition of the categories into K groups in the case of two class values, with the use of a dynamic programming algorithm of time complexity I^2 . In the general case of more than two class values, there is no algorithm to find the optimal grouping with K groups, apart from exhaustive search. However, Chou (1991) has proposed an approach based on K-means that allows finding a locally optimal partition of the categories into K groups. Decision tree algorithms often manage the grouping problem with a greedy heuristic based on a bottom-up classification of the categories. The algorithm starts with single category groups and then searches for the best merge between groups. The process is reiterated until no further merge can improve the grouping criterion. The CHAID algorithm (Kass, 1980) uses this greedy approach with a criterion close to ChiMerge (Kerber, 1991). The best merges are searched by minimizing the chi-square criterion applied locally to two categories: they are merged if they are statistically similar. The ID3 algorithm (Quinlan, 1986) uses the information gain criterion to evaluate categorical attributes, without any grouping. This criterion tends to favor attributes with numerous categories and Quinlan (1993) proposed in C4.5 to exploit the gain ratio criterion, by dividing the information gain by the entropy of the categories. The chi-square criterion has also been applied globally on the whole set of categories, with a normalized version of the chi-square value (Ritschard et al., 2001) such as the Cramer's V or the Tschuprow's T, in order to compare two different-size partitions.

In this paper, we present a new grouping method called MODL, which results from a similar approach as that of the MODL discretization method (Boullé, 2004c). This method is founded on a Bayesian approach to find the most probable grouping model given the data. We first define a general family of grouping models, and second propose a prior distribution on this model space. This leads to an evaluation criterion of groupings, whose minimization defines the optimal grouping. We use a greedy bottom-up algorithm to optimize this criterion. The method starts the grouping from the elementary single value groups. It evaluates all merges between groups, selects the best one according to the MODL criterion and iterates this process. As the grouping problem has been turned into a minimization problem, the method automatically stops merging groups as soon as the evaluation of the resulting grouping does not decrease anymore. Additional preprocessing and post-optimization steps are proposed in order to improve the solutions while keeping a super-linear optimization time. Extensive experiments show that the MODL method produces high quality groupings in terms of compactness, robustness and accuracy.

The remainder of the paper is organized as follows. Section 2 describes the MODL method. Section 3 proceeds with an extensive experimental evaluation.

2 The MODL Grouping Method

In this section, we present the MODL approach which results in a Bayes optimal evaluation criterion of groupings and the greedy heuristic used to find a near-optimal grouping.

2.1 Presentation

In order to introduce the issues of grouping, we present in Figure 1 an example based on the Mushroom UCI data set (Blake and Merz, 1998). The class attribute has two values: EDIBLE and POISONOUS. The 10 categorical values of the explanatory attribute CapColor are sorted by decreasing frequency; the proportions of the class values are reported for each explanatory value. Grouping the categorical values does not make sense in the unsupervised context. However, taking the class attribute into account introduces a metric between the categorical values. For example, looking at the proportions of their class values, the YELLOW cap looks closer from the RED cap than from the WHITE cap.

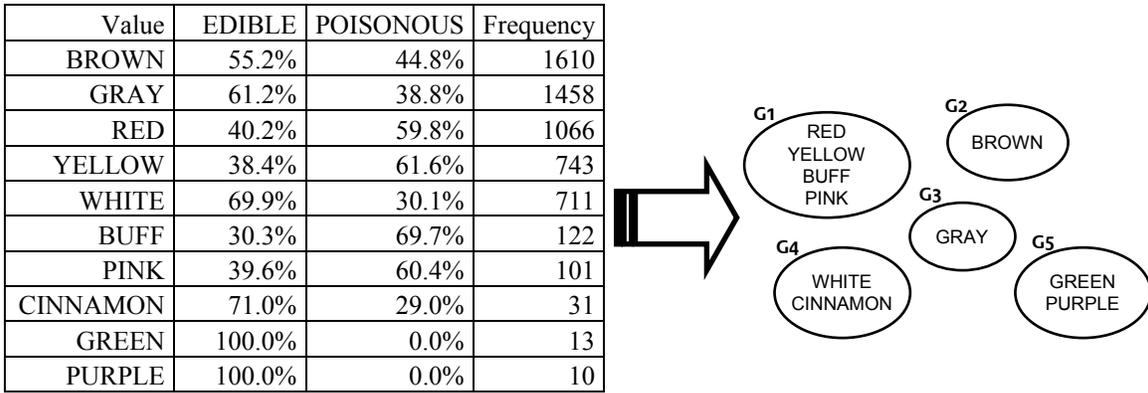


Figure 1. Example of a grouping of the categorical values of the attribute CapColor of data set Mushroom

In data preparation for supervised learning, the problem of grouping is to produce the smallest possible number of groups with the slightest decay of information concerning the class values. In Figure 1, the values of CapColor are partitioned into 5 groups. The BROWN and GRAY caps are kept into 2 separate groups, since their relatively small difference of proportions of the class values is significant (both categorical values have important frequencies). On the opposite, the BUFF cap is merged with the RED, YELLOW and PINK caps: the frequency of the BUFF cap is not sufficient to make a significant difference with the other values of the group.

The issue of a good grouping method is to find a good trade-off between information (as many groups as possible with discriminating proportions of class values) and reliability (the class information learnt on the train data should be a good estimation of the class information observed on the test data). Producing a good grouping is harder with large numbers of values since the risk of overfitting the data increases. In the limit situation where the number of values is the same as the number of instances, overfitting is obviously so important that efficient grouping methods should produce one single group, leading to the elimination of the attribute. In real applications, there are some domains that require grouping of the categorical attributes. In marketing applications for example, attributes such as Country, State, ZipCode, FirstName, ProductID usually hold many different values. Preprocessing these attributes is critical to produce efficient classifiers.

2.2 Definition of a Grouping Model

The objective of the grouping process is to induce a set of groups from the set of values of a categorical explanatory attribute. The data sample consists of a set of instances described by pairs of values: the explanatory value and the class value. The explanatory values are categorical: they can be distinguished from each other, but they cannot *naturally* be sorted. We propose in Definition 1 the following formal definition of a grouping model. Such a model is a pattern that describes both the partition of the categorical values into groups and the proportions of the class values in each group.

Definition 1: A *standard* grouping model is defined by the following properties:

1. the grouping model allows to describe a partition of the categorical values into groups,
2. in each group, the distribution of the class values is defined by the frequencies of the class values in this group.

Such a grouping model is called a SGM model.

Notations:

- n : number of instances
- J : number of classes
- I : number of categorical values
- n_i : number of instances for value i
- n_{ij} : number of instances for value i and class j
- K : number of groups
- $k(i)$: index of the group containing value i
- n_k : number of instances for group k
- n_{kj} : number of instances for group k and class j

The purpose of a grouping model is to describe the distribution of the class attribute conditionally to the explanatory attribute. All the information concerning the explanatory attribute, such as the size of the data set, the number of explanatory values and their distribution might be used by a grouping model. The input data can be summarized knowing n , J , I and n_i . The grouping model has to describe the partition of the explanatory values into groups and the distribution of the class values in each group. A SGM grouping model is completely defined by the parameters $\{ K, \{k(i)\}_{1 \leq i \leq I}, \{n_{kj}\}_{1 \leq k \leq K, 1 \leq j \leq J} \}$.

For example, in Figure 1, the input data consists of $n=5865$ instances (size of the train data set used in the sample), $I=10$ categorical values and $J=2$ class values. The n_i parameters represents the counts of the categorical values (for example, $n_1=1610$ for the BROWN CapColor).

The grouping model pictured in Figure 1 is defined by $K=5$ groups, the description of the partition of the 10 values into 5 groups (for example, $k(1)=2$ since the BROWN CapColor belongs to the G2 group) and the description of the distribution of the class values in each group. This last set of parameters ($\{n_{kj}\}_{1 \leq k \leq 5, 1 \leq j \leq 2}$) corresponds to the counts in the contingency table of the grouped attribute and class attribute.

2.3 Evaluation of a Grouping Model

In the Bayesian approach, the best model is found by maximizing the probability $P(\text{Model}/\text{Data})$ of the model given the data. Using Bayes rule and since the probability $P(\text{Data})$ is constant under varying the model, this is equivalent to maximize $P(\text{Model})P(\text{Data}/\text{Model})$. For a detailed presentation on Bayesian theory and its applications to model comparison and hypothesis testing, see for example (Bernardo and Smith, 1994; Kass and Raftery, 1995).

Once a prior distribution of the models is fixed, the Bayesian approach allows to find the optimal model of the data, provided that the calculation of the probabilities $P(\text{Model})$ and $P(\text{Data}/\text{Model})$ is feasible. We define below a prior which is essentially a uniform prior at each stage of the hierarchy of the model parameters. We also introduce a strong hypothesis of independence of the distribution of the class values. This hypothesis is often assumed (at least implicitly) by many grouping methods that try to merge similar groups and separate groups with significantly different distributions of class values. This is the case for example with the CHAID grouping method (Kass, 1980), which merges two adjacent groups if their distributions of class values are statistically similar (using the chi-square test of independence).

Definition 2: The following distribution prior on SGM models is called the three-stage prior:

1. the number of groups K is uniformly distributed between 1 and I ,
2. for a given number of groups K , every division of the I categorical values into K groups is equiprobable,
3. for a given group, every distribution of class values in the group is equiprobable,
4. the distributions of the class values in each group are independent from each other.

Owing to the definition of the model space and its prior distribution, Bayes formula is applicable to exactly calculate the prior probabilities of the model and the probability of the data given the model. Theorem 1, proven in Appendix A, introduces a Bayes optimal evaluation criterion.

Theorem 1: A SGM model distributed according to the three-stage prior is Bayes optimal for a given set of categorical values if the value of the following criterion is minimal on the set of all SGM models:

$$\log(I) + \log(B(I, K)) + \sum_{k=1}^K \log(C_{n_k+J-1}^{J-1}) + \sum_{k=1}^K \log(n_k! / n_{k,1}! n_{k,2}! \dots n_{k,J}!).$$

C is the combinatorial operator. $B(I, K)$ is the number of divisions of the I values into K groups (with eventually empty groups). When $K=I$, $B(I, K)$ is the Bell number. In the general case, $B(I, K)$ can be written as a sum of Stirling numbers of the second kind $S(I, k)$:

$$B(I, K) = \sum_{k=1}^K S(I, k).$$

$S(I, k)$ stands for the number of ways of partitioning a set of I elements into k nonempty sets.

The first term of the criterion in equation 1 stands for the choice of the number of groups, the second term for the choice of the division of the values into groups and the third term for the choice of the class distribution in each group. The last term encodes the probability of the data given the model.

There is a subtlety in the three-stage prior, where choosing a grouping with K groups incorporates the case of potentially empty groups. The partitions are thus constrained to have at most K groups instead of exactly K groups. The intuition behind the choice of this prior is that if K groups are chosen and if the categorical values are dropped in the groups independently from each other, empty groups are likely to appear. We present below two theorems (proven in Appendix A) that bring a more theoretical justification of the choice of the prior. These theorems are no longer true when the prior is to have partition containing exactly K groups.

Definition 3: A categorical value is *pure* if it is associated with a single class.

Theorem 2: In a Bayes optimal SGM model distributed according to the three-stage prior, two pure categorical values having the same class are necessary in the same group.

This brings an intuitive validation of the MODL approach. Furthermore, grouping algorithms can exploit this property in a preprocessing step and considerably reduce their overall computational complexity.

Theorem 3: In a SGM model distributed according to the three-stage prior and in the case of two classes, the Bayes optimal grouping model consists of a single group when each instance has a different categorical value.

This provides another validation of the MODL approach. Building several groups in this case would reflect an over-fitting behavior.

Conjecture 1: In a Bayes optimal SGM model distributed according to the three-stage prior and in the case of two classes, any categorical value whose class proportion is between the class proportions of two categorical values belonging to the same group necessary belongs to this group.

This conjecture has been proven for other grouping criterion such as Gini (Breiman, 1984) or Kolmogorov-Smirnov (Asseraf, 2000) and experimentally validated in extensive experiments for the MODL criterion. It will be considered as true in the following. The grouping algorithms, such as greedy bottom-up merge algorithms, can take benefit from this conjecture. Once the categorical values have been sorted by decreasing proportion of the class, the number of potentially interesting value merges is reduced to $(I-1)$ instead of $I(I-1)/2$.

2.4 Optimization of a Grouping Model

Once the optimality of an evaluation criterion is established, the problem is to design a search algorithm in order to find a grouping model that minimizes the criterion. In this section, we present a greedy bottom-up merge heuristic enhanced with several preprocessing and post-optimization algorithms whose purpose is to achieve a good trade-off between the time complexity of the search algorithm and the quality of the groupings.

2.4.1 Greedy Bottom-Up Merge Heuristic

In this section, we present a standard greedy bottom-up heuristic. The method starts with initial single value groups and then searches for the best merge between groups. This merge is performed if the MODL evaluation criterion of the grouping decreases after the merge and the process is reiterated until not further merge can decrease the criterion.

The algorithm relies on the $O(I)$ marginal counts, which require one $O(n)$ scan of the data set. However, we express the complexity of the algorithm in terms of n (rather than in terms of I), since the number of categorical values can reach $O(n)$ in the worst case. With a straightforward implementation of the algorithm, the method runs in $O(n^3)$ time (more precisely $O(n+I^3)$). However, the method can be optimized in $O(n^2 \log(n))$ time owing to an algorithm similar to that presented in (Boullé, 2004a). The algorithm is mainly based on the additivity of the evaluation criterion. Once a grouping is evaluated, the value of a new grouping resulting from the merge between two adjacent groups can be evaluated in a single step, without scanning all the other groups. Minimizing the value of the groupings after the merges is the same as maximizing the related variation of value Δ value. These Δ values can be kept in memory and sorted in a maintained sorted list (such as an AVL binary search tree for example). After a merge is completed, the Δ values need to be updated only for the new group and its adjacent groups to prepare the next merge step.

Optimized greedy bottom-up merge algorithm:

- Initialization
 - Create an elementary group for each value: $O(n)$
 - Compute the value of this initial grouping: $O(n)$
 - Compute the Δ values related to all the possible merges of 2 values: $O(n^2)$
 - Sort the possible merges: $O(n^2 \log(n))$
- Optimization of the grouping: repeat the following steps (at most n steps)
 - Search for the best possible merge: $O(1)$
 - Merge and continue if the best merge decreases the grouping value
 - Compute the Δ values of the remaining group merges adjacent to the best merge: $O(n)$
 - Update the sorted list of merges: $O(n \log(n))$

In the case of two classes, the time complexity of the greedy algorithm can be optimized down to $O(n \log(n))$ owing to conjecture 1.

2.4.2 Preprocessing

In the general case, the computational complexity is not compatible with large real databases, when the categorical values becomes too numerous. In order to keep a super-linear time complexity, the initial categorical values can be preprocessed into a new set of values of cardinality $I' \leq \sqrt{n}$.

A first straightforward preprocessing step is to merge pure values having the same class. This step is compatible with the optimal solution (see Theorem 2).

A second preprocessing step consists in building J groupings for each "one class against the others" sub-problems. This require $O(J n \log(n))$ time. The subparts of the groups shared by all the J groupings can easily be identified and represent very good candidate subgroups of the global grouping problem. The number of these subparts is usually far below the number of initial categorical values and helps achieving a reduced sized set of preprocessed values.

A third preprocessing step can be applied when the number of remaining preprocessed values is beyond \sqrt{n} . The values can be sorted by decreasing frequencies and the exceeding infrequent values can be unconditionally grouped into J groups according to their majority class. This last step is mandatory to control the computational complexity of the algorithm. However, experiments show that this last step is rarely activated in practice.

2.4.3 Post-Optimizations

The greedy heuristic may fall in a local optimum, so that time efficient post-optimizations are potentially useful to improve the quality of the solution. Since the evaluation criterion is Bayes optimal, spending more computation time is meaningful.

A first post-optimization step consists in forcing the merges between groups until a single terminal group is obtained and to keep the best encountered grouping. This helps escaping local optima and requires $O(n \log(n))$ computation time. Furthermore, in the case of noisy attribute where the optimal grouping consists of a single group, this heuristic guarantees to find the optimal solution.

A second post-optimization step consists in evaluating every move of a categorical value from one group to another. The best moves are performed as long as they improve the evaluation criterion. This process is similar to the K-means algorithm where each value is attracted by its closest group. It converges very quickly, although this cannot be proved theoretically.

A third post-optimization step is a look-ahead optimization. The best merge between groups is simulated and post-optimized using the second step algorithm. The merge is performed in case of improvement. This algorithm looks similar to the initial greedy merge algorithm, except that it starts from a very good solution and incorporates an enhanced post-optimization. Thus, this additional post-optimization is usually triggered for only one or two extra merges.

3 Experiments

In our experimental study, we compare the MODL grouping method with other supervised grouping algorithms. In this section, we introduce the evaluation protocol, the alternative evaluated grouping methods and the evaluation results on artificial and real data sets. Finally, we present the impact of grouping as a preprocessing step to the Naïve Bayes classifier.

3.1 Presentation

In order to evaluate the intrinsic performance of the grouping methods and eliminate the bias of the choice of a specific induction algorithm, we use a protocol similar as (Boullé, 2004b), where each grouping method is considered as an elementary inductive method which predicts the distribution of the class values in each learned groups.

The grouping problem is a bi-criteria problem that tries to compromise between the predictive quality and the number of groups. The optimal classifier is the Bayes classifier: in the case of an univariate classifier based on a single categorical attribute, the optimal grouping is to do nothing,

i.e. to build one group per categorical value. In the context of data preparation, the objective is to keep most of the class conditional information contained in the attribute while decreasing the number of values. In the experiments, we collect both the number of groups and the predictive quality of the grouping. The number of groups is easy to calculate. The quality of the estimation of the class conditional information hold by each group is more difficult to evaluate.

We choose not to use the accuracy criterion because it focuses only on the majority class value and cannot differentiate correct predictions made with probability 1 from correct predictions made with probability slightly greater than 0.5. Furthermore, many applications, especially in the marketing field, rely on the scoring of the instances and need to evaluate the probability of each class value. In the case of categorical attributes, we have the unique opportunity of observing the class conditional distribution on the test data set: for each categorical value in the test data set, the observed distribution of the class values can be estimated by counting. The grouping methods allow to induce the class conditional distribution from the train data set: for each learnt group on the train data set, the learnt distribution of the class values can be estimated by counting. The objective of grouping is to minimize the distance between the learnt distribution and the observed distribution. This distance can be evaluated owing to a divergence measure, such as the Kullback-Leibler divergence, the chi-square divergence or the Hellinger coefficient. In our experiments, we choose to evaluate this distance using the Kullback-Leibler divergence (Kullback, 1968).

The MODL grouping methods exploits a space of class conditional distribution models (the SGM models) and searches the most probable model given the train data. It is noteworthy that no loss function is optimized in this approach: neither the classification accuracy is optimized nor the Kullback-Leibler divergence (which would require to divide the train data set into train data and validation data). The MODL method exploits all the available train data to build its grouping model.

The evaluation conducted in the experiments focuses on the quality of the groupings (size and Kullback-Leibler divergence criterions) as a preprocessing method for data mining. It is interesting to examine whether optimizing the posterior probability of a grouping model (on the basis on the three-stage prior) leads to high-quality groupings.

3.2 The Evaluation Protocol

The predictive quality of the groupings is evaluated owing to the Kullback-Leibler divergence (Kullback, 1968) applied to compare the distribution of the class values estimated from the train data set with the distribution of the class values observed on the test data set.

Let n' be the number of instances, n'_i be the number of instances for value i and n'_{ij} the number of instances for value i and class j on the test data set.

For a given categorical value i , let p_{ij} be the probability of the j^{th} class value estimated on the train data set (on the basis of the group containing the categorical value), and q_{ij} be the probability of the j^{th} class value observed on the test data set (using directly the categorical value). The q_{ij} probabilities are estimated with the Laplace's estimator in order to deal with zero values. We get

$$p_{ij} = \frac{n_{k(i)j}}{n_{k(i)}} \quad \text{and} \quad q_{ij} = \frac{n'_{ij} + 1}{n'_i + J}.$$

The mean of the Kullback-Leibler divergence on the test data set is

$$D(p||q) = \sum_{i=1}^I \frac{n'_i}{n'} \sum_{j=1}^J p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

In a first experiment, we compare the behavior of the evaluated grouping method on synthetic data sets, where the ideal grouping pattern is known in advance. In the second experiments, we

use real data sets to compare the grouping methods considered as univariate classifiers. In a third experiment, we use the same data sets to evaluate the results of Naïve Bayes classifiers using the grouping methods to preprocess the categorical attributes. In this last experiment, the results are evaluated using the classification accuracy both on train data sets and on test data sets.

Data Set	Continuous Attributes	Categorical Attributes	Size	Class Values	Majority Accuracy
Adult	7	8	48842	2	76.07
Australian	6	8	690	2	55.51
Breast	10	0	699	2	65.52
Crx	6	9	690	2	55.51
Heart	10	3	270	2	55.56
HorseColic	7	20	368	2	63.04
Ionosphere	34	0	351	2	64.10
Mushroom	0	22	8416	2	53.33
TicTacToe	0	9	958	2	65.34
Vehicle	18	0	846	4	25.77
Waveform	40	0	5000	3	33.84
Wine	13	0	178	3	39.89

Table 1. Data sets

We gathered 12 data sets from U.C. Irvine repository (Blake and Merz, 1998), each data set has at least a few tenths of instances for each class value and some categorical attributes with more than two values. In order to increase the number of categorical attributes candidate for grouping, the continuous attributes have been discretized in a preprocessing step with a 10 equal-width unsupervised discretization. The 12 data sets comprising 230 attributes are described in Table 1; the last column corresponds to the accuracy of the majority class.

The categorical attributes in these data sets hold less than 10 values on average (from an average 3 values in the TicTacToe attributes to about 20 values in the HorseColic attributes). In order to perform more discriminating experiments, we use a second collection of data sets containing all the cross-products of the attributes. In this "bivariate" benchmark, the 12 data sets contain 2614 categorical attributes holding 55 values on average.

3.3 The Evaluated Methods

The grouping methods studied in the comparison are:

- MODL, the method described in this paper,
- Khiops (Boulle, 2004b),
- BIC (Ritschard, 2003),
- CHAID (Kass, 1980),
- Tschuprow (Ritschard et al., 2001),
- Gain Ratio (Quinlan, 1993).

All these methods are based on a greedy bottom-up algorithm that iteratively merges the categories into groups, and automatically determines the number of groups in the final partition of the categories. The MODL method is based on a Bayesian approach and incorporates preprocessing and post-optimization algorithms. The Khiops, CHAID, Tschuprow and BIC methods use the chi-square statistics in different manner. The Gain Ratio method is based on entropy.

The CHAID method is the grouping method used in the CHAID decision tree classifier (Kass, 1980). It applies the chi-square criterion locally to two categorical values in the

contingency table, and iteratively merges the values as long as they are statistically similar. The significance level is set to 0.95 for the chi-square threshold. The Tschuprow method is based on a global evaluation of the contingency table, and uses the Tschuprow's T normalization of the chi-square value to evaluate the partitions. The Khiops method also applies the chi-square criterion on the whole contingency table, but it evaluates the partition using the confidence level related to the chi-square criterion. Furthermore, the Khiops method provides a guaranteed resistance to noise: any categorical attribute independent from the class attribute is grouped in a single terminal group with a user defined probability. This probability is set to 0.95 in the experiments. The BIC method is based on the deviance G^2 statistics, which is a chi-square statistics. It exploits a Bayesian information criterion (Schwarz, 1978) to select the best compromise model between fit and complexity. The Gain Ratio method is the methods used in the C4.5 decision tree classifier (Quinlan, 1993). The gain ratio criterion attempts to find a trade-off between the information on the class values (information gain) and the complexity of the partition (the split information) by dividing the two quantities.

We have re-implemented these alternative grouping approaches in order to eliminate any variance resulting from different cross-validation splits.

3.4 The Artificial Data Sets Experiments

Using artificial data sets allows controlling the distribution of the explanatory values and of the class values. The evaluation of the groupings learned on train data sets can thus be optimal, without requiring any test data set. In the case of grouping, an artificial data set containing one categorical attribute and one class attribute is completely specified by the following parameters:

I : number of categorical values,

J : number of classes,

p_i , $1 \leq i \leq I$: probability distribution of the categorical values,

p_{ji} , $1 \leq j \leq J$, $1 \leq i \leq I$: probability distribution of the class values conditionally to the categorical values.

3.4.1 The Noise Pattern

The purpose of this experiment is to compare the robustness of the grouping methods. The *noise pattern* data set consists of an explanatory categorical attribute independent from the class attribute. The explanatory attribute is uniformly distributed ($p_i=1/I$) and the class attribute consists of two equidistributed class values ($p_{ji}=1/2$). We use randomly generated train samples of size 1000 and perform the experiment 1000 times, for different numbers of categorical values. In the case of independence, the optimal number of groups is 1. In Figure 2, we report the mean of the number of unnecessary groups ($K-1$) and of the Kullback-Leibler divergence between the estimated class distribution and the true class distribution.

The results demonstrate the effectiveness of the Kullback-Leibler divergence to evaluate the quality of a grouping. In the case of attribute independence, the classification accuracy is uniformly equal to 50% whatever the number of groups: it is useless as an evaluation criterion. The most comprehensible grouping consists of a single group whereas the worst one is to build one group per value. The Kullback-Leibler divergence is able to exploit this by a better evaluation of the true class distribution in the most frequent groups. Thus, building separate groups that could be merged leads to a poorer estimation of the class distribution.

The CHAID method creates more and more groups when the number of categorical values increases. This translates by a quickly decreasing quality of estimation of the class distribution, as pictured in Figure 2.

The BIC method performs better than the CHAID method when the number of categorical values is small. When this number of values increases, the BIC and CHAID methods perform similarly.

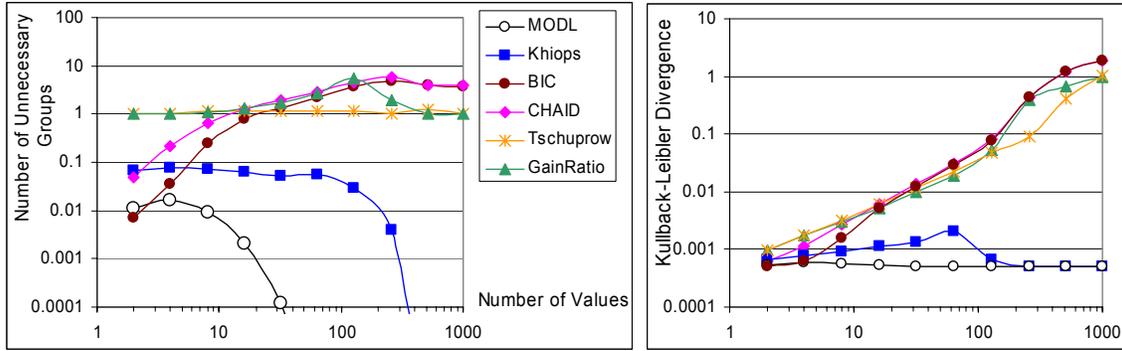


Figure 2. Mean of the number of unnecessary groups ($K-1$) and of the Kullback-Leibler divergence of the groupings of an explanatory attribute independent from the class attribute

The GainRatio method is constrained to produce at least 2 groups. It overfits the train data as soon as the number of categorical values exceeds a few tens.

The Tschuprow method is also constrained to produce at least 2 groups. It builds almost systematically exactly two groups. A closer look at the Tschuprow criterion shows that the criterion can reach its theoretical bound only when the contingency table is square, meaning that the number of groups is exactly equal to the number of class values. Additional experiments with different numbers of class values (not reported in this paper) confirm this bias. Although the number of groups is constant under varying the number of categorical values, the estimation of the class distribution worsens with higher number of values. This is explained since less frequent values lead to a less reliable estimation of the class probabilities.

The Khiops method is designed to build one single group with probability 0.95, when the explanatory attribute and the class attribute are independent. This is confirmed by the experiment up to about 100 categorical values. Above this number of values, the Khiops method systematically builds one group for the reason that it unconditionally groups the least frequent values in a preprocessing step. The objective of this preprocessing step is to improve the reliability of the confidence level associated with the chi-square criterion, by constraining every cell of the contingency table to have an expected value of at least 5.

The MODL method builds one single group almost always, and the proportion of multi-groups partitions decreases sharply with the number of categorical values. The experiments have been performed 100000 times for the MODL method. Above 50 values, no grouping (out of 100000) contains more than one group.

3.4.2 The Mixture Pattern

The objective of this experiment is to evaluate the sensibility of the grouping methods. The *mixture pattern* data set consists of an explanatory categorical attribute distributed according to several mixtures of class values. The explanatory attribute is uniformly distributed ($p_i=1/I$) and the class attribute consists of four equidistributed class values ($p_{j/i}=1/4$). We designed 8 artificial groups corresponding to 8 different mixtures of class values. In each group, one of the class values is the majority class (with probability 0.50 or 0.75) and the other three classes are equidistributed, as pictured in Figure 3.

We use randomly generated train samples of size 10000 and perform the experiment 1000 times, for different numbers of categorical values. Due to the quadratic complexity of the algorithms, the experiment was conducted up to 2000 categorical values, except for the MODL algorithm that has a super-linear complexity. In Figure 4, we report the mean of the number of groups and of the Kullback-Leibler divergence between the estimated class distribution and the true class distribution.

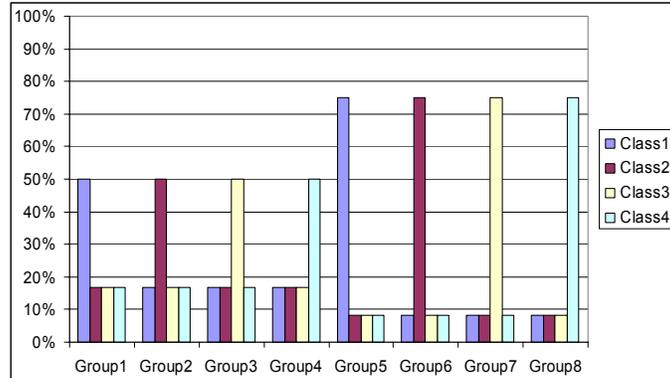


Figure 3. Class distribution in 8 artificial group and 4 class values; the categorical values are uniformly distributed on the 8 groups

The CHAID method overfits the training data by creating too many groups, even more than in the case of independence since the number of class values is now 4. The BIC method manages to find the optimal number of groups when the number of categorical values is below 100. Above this threshold, it overfits the training data and exhibits a behavior similar to that of the CHAID method. The behavior of the GainRatio method is unexpected. For small numbers of categorical values, it builds a constant number of groups equal to the number of class values, and beyond one hundred values, the number of groups raises sharply. The Tschuprow method is so strongly biased that it always produces exactly 4 groups. The Khiops method benefits from its robustness and correctly identifies the 8 artificial groups as long as the frequencies of the categorical values are sufficient. Beyond about 400 values, the minimum frequency constraint of the Khiops algorithms become active and the number of groups falls down to 1. The MODL method almost always builds optimal groupings with the correct number of groups. When the number of categorical values becomes large (about 500, *i.e.* an average frequency of 40 per value), there is a transition in the behavior of the algorithm, that produces only 4 groups instead of 8. The frequency of the categorical values is no longer sufficient to discriminate 8 types of class distributions. When the number of class values increases again (beyond 2000, *i.e.* an average frequency of 5 per value), there is a second transition and the MODL method builds one single group.

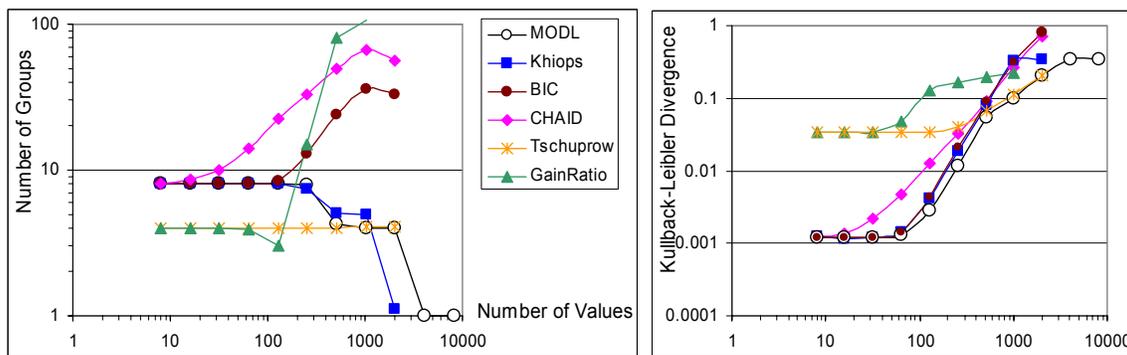


Figure 4. Mean of the number of groups and of the Kullback-Leibler divergence of the groupings of an explanatory attribute distributed in 8 mixtures of class values

To summarize, the noise and mixture pattern experiments are a convenient way to characterize each grouping method with their bias, robustness, sensibility and limits. The experiments show the interest of using the Kullback-Leibler divergence to evaluate the quality of

the groupings. This criterion looks well suited to evaluate the groupings in real data sets, where the true class distribution is unknown.

The CHAID method exhibits an overfitting behavior, which decays when the number of categorical values or class values increases. The BIC method finds the correct number of groups when the categorical values are not too numerous and then overfits the training data. The Tschuprow is strongly biased in favor of numbers of groups equal to numbers of classes. The GainRatio exhibits a varying biased and overfitting behavior according to the distribution of the train data. The Khiops method is robust but suffers from a lack of sensibility when the categorical values are too numerous, due to its minimum frequency constraint.

The MODL method builds groupings that are optimum, the most probable groupings given the train data. It is the only evaluated method that retrieves the exact number of groups both in case of noise data and of informative data.

3.5 The Real Data Sets Experiments

The goal of this experiment is to evaluate the intrinsic performance of the grouping methods, without the bias of the choice of a specific induction algorithm. The grouping are performed on all the attributes of the UCI data sets presented in Table 1, using a stratified tenfold cross-validation. As the purpose of the experiments is to evaluate the grouping methods according to the way they compromise between the quality and the size of the groupings, we also added three basic grouping methods for comparison reasons. The first method named NoGrouping builds one group per categorical value: it is the least biased method for estimating the distribution of the class values at the expense of the highest number of groups. The second method called ExhaustiveCHAID (SPSS, 2001) is a version of the CHAID method that merges similar pairs continuously until only a single pair remains. We added a similar method ExhaustiveMODL, which allows comparing the two methods when they are constrained to build the same number of groups.

During the experiments, we collect the number of groups and the Kulback-Leibler divergence between the class distribution estimated on train data sets and the class distribution observed on test data sets. For each grouping method, this represents 2300 measures for the univariate analysis (230 attributes) and 26140 measures for the bivariate analysis (2614 pairs of attributes). All these results are summarized across the attributes of the data sets owing to means, in order to provide a gross estimation of the relative performances of the methods. We report the mean of the number of groups and of the Kullback-Leibler divergence for the univariate and bivariate analysis in Figure 5. For the Kullback-Leibler divergence, we use geometric means normalized by the result of the NoGrouping method in order to focus on the ratios of predictive performance between tested methods. The gray line highlights the Pareto curve of the results obtained by the grouping methods.

As expected, the NoGrouping method obtains the best results in term of predictive quality, at the expense of the worst number of groups. However, in the univariate analysis, the Khiops, BIC, CHAID and MODL methods reach almost the same quality with far less groups. The Tschuprow method is hampered by its bias in favor of number of groups equal to the number of class values, so that its performance are not better that those of groups equal to the number of class values, ExhaustiveCHAID and ExhaustiveMODL. The GainRatio method is dominated by the other methods. The bivariate analysis is much more selective. The results follow the same trend with sharper differences between the methods. The Khiops method underfits the data because of its minimum frequency constraint, while the CHAID method suffers from its lack of overfitting control by producing too many groups and degrading its predictive performance. Although its criterion incorporates a complexity penalty, the BIC method builds too many groups and overfits the data.

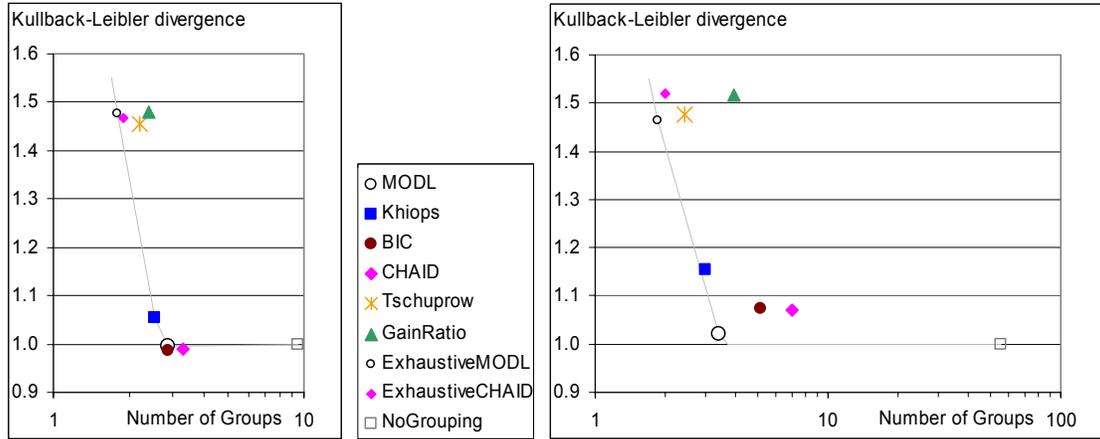


Figure 5. Mean of the number of groups and of the Kullback-Leibler divergence of the groupings performed on the UCI data sets, in univariate analysis (on the left) and bivariate analysis (on the right)

The MODL method gets the lowest number of group without discarding the predictive quality. It manages to reduce the number of categorical values by one order of magnitude while keeping the best estimate of the class conditional probability.

3.6 Impact of Groupings on the Naïve Bayes Classifier

The aim of this experiment is to evaluate the impact of grouping methods on the Naïve Bayes classifier. The Naïve Bayes classifier (Langley et al., 1992) assigns the most probable class value given the explanatory attributes values, assuming conditional independence between the attributes for each class value. The Naïve Bayes classifier is a very simple technique that does not require any parameter setting, which removes the bias due to parameter tuning. It performs surprisingly well on many data sets (Dougherty et al., 1995; Hand and Yu, 2001) and is very robust. High quality grouping tend to increase the frequency in each group and to decrease the variance in the estimation of the conditional class density. It is interesting to examine whether this can benefit to classifiers, even to the Naïve Bayes classifier which is particularly unsophisticated.

The evaluation of probabilities for numeric attributes owing to discretization has already been discussed in the literature (Dougherty et al., 1995; Hsu et al, 2003; Yang and Webb, 2003). Experiments have shown that even a simple Equal Width discretization with 10 bins brings superior performances compared to the assumption using the Gaussian distribution. On the opposite, the probabilities for categorical attribute are estimated using the Laplace's estimator directly on the categorical values, without any preprocessing such as grouping. It is interesting to study whether grouping could produce a more robust estimation of the class distributions and enhance the performance of the Naïve Bayes classifier. The experiment is performed on the 12 data sets presented in Table 1, using the univariate (all categorical attributes) and bivariate (all pairwise interactions of categorical attributes) sets of data sets. A Student's test at the 5% confidence level is performed between the MODL grouping method and the other methods to determine whether the differences of performance are significant. According to (Dietterich, 1998), the McNemar's test is more reliable, but it does not assess the effect of varying the training set. However, these statistical tests "must be viewed as approximate, heuristic tests, rather than as rigorously correct statistical methods" (Dietterich, 1998). Table 2 reports a summary of the test accuracy and robustness results, using the mean of the data set results, the average rank of each method and the number of significant wins and losses of the MODL method.

	Test accuracy						Robustness (Test acc. / Train acc.)					
	Univariate data sets			Bivariate data sets			Univariate data sets			Bivariate data sets		
	Mean	Rank	Wins	Mean	Rank	Wins	Mean	Rank	Wins	Mean	Rank	Wins
MODL	84.8%	2.6		85.8%	2.8		98.6%	3.0		96.8%	4.2	
Khiops	84.1%	3.4	3/1	83.7%	4.7	4/0	98.6%	3.8	1/0	97.2%	2.5	0/3
BIC	83.2%	4.2	4/0	84.9%	3.2	3/0	96.1%	5.2	3/0	93.9%	6.7	7/0
CHAID	83.3%	3.7	2/0	84.7%	4.0	2/1	96.1%	5.2	1/0	93.6%	7.3	7/0
Tschuprow	82.8%	6.4	7/0	83.3%	6.6	6/0	96.6%	5.2	1/0	93.8%	5.5	3/0
GainRatio	81.5%	6.1	5/1	82.9%	5.8	4/0	95.4%	6.0	1/0	93.0%	5.7	5/0
ExMODL	83.4%	5.2	5/0	84.6%	4.7	4/0	98.5%	3.8	0/0	97.1%	2.7	0/2
ExCHAID	81.9%	7.4	7/0	83.2%	5.7	6/0	96.2%	5.8	2/0	94.6%	3.5	2/2
NoGrouping	84.0%	4.1	4/0	84.6%	5.1	4/0	97.3%	6.4	3/0	94.0%	6.2	6/0

Table 2. Summary of the test accuracy and robustness results (mean, average rank and number of wins/losses) of the Naïve Bayes classifier on the UCI data sets, in univariate analysis and bivariate analysis

The results look consistent on the three indicators and show that the MODL method dominates the other methods on the test accuracy criterion. The mean results are pictured in Figure 6 with the classification accuracy reported both on train and test data sets, in order to visualize the train and test accuracy of the methods in a two criteria-analysis. The thick gray line on the diagonal represents the asymptotic best achievable robustness of the methods. The thin gray line highlights the Pareto curve in the two-criteria analysis between robustness and test accuracy.

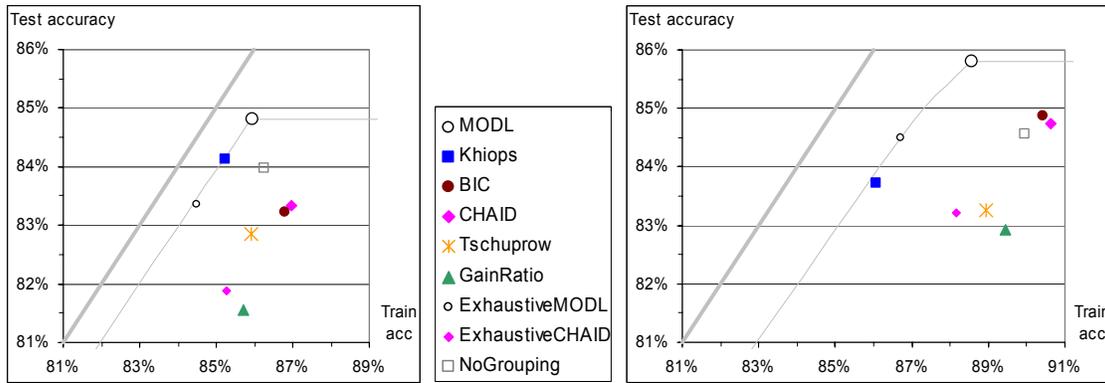


Figure 6. Mean of the train accuracy and test accuracy of the Naïve Bayes classifier on the UCI data sets, in univariate analysis (on the left) and bivariate analysis (on the right)

Most methods do not perform better than the NoGrouping method. This probably explains why the Naïve Bayes classifiers do not make use of groupings in the literature. The MODL method clearly dominates all the other methods, owing to the quality of its groupings. The resulting Naïve Bayes classifier is both the more robust one (together with Khiops) and the more accurate on test data sets. Another important aspect learnt from this experiment is the overall gain in test accuracy when the pairs of attributes are used. The bivariate analysis allows to investigate simple interactions between attributes and to go beyond the limiting independence assumption of the Naïve Bayes classifier. Although this degrades the robustness (because of a decrease in the frequency of the categorical values), this enhances the test accuracy. From univariate to bivariate analysis, the MODL method achieves an increase in test accuracy about twice and a decay in robustness approximately half that of the reference NoGrouping method.

Compared to the NoGrouping method, the MODL method is the only evaluated grouping method that improves the test accuracy of the Naïve Bayes classifier. The most noticeable effect of using the MODL method is a drastic improvement of the robustness.

4 Conclusion

When categorical attributes contain few values, typically less than 10 values, using a grouping method is not required. As the number of values increases (which is common in marketing applications), preprocessing the categorical attributes becomes attractive in order to improve the performance of classifiers. The issue of grouping methods is to reduce the number of groups of values while maintaining the conditional class information.

The MODL grouping method exploits the precise definition of a family of grouping models with a general prior. This provides a new evaluation criterion which is minimal for the Bayes optimal grouping, *i.e.* the most probable grouping given the data sample. An optimization heuristics including preprocessing and post-optimizations is proposed in this paper to optimize the grouping with super-linear time complexity. This algorithm manages to efficiently find high quality groupings.

Extensive evaluations both on real and synthetic data indicate notable performances for the MODL method. It is time efficient and does not require any parameter setting. It builds groupings that are both robust and accurate. The more valuable characteristic of the MODL method is probably the understandability of the groupings. Although understandability is hard to evaluate, the method is theoretically founded to produce correct explanations of the explanatory categorical attributes on the basis of the partition of their values, and even the most probable "grouping based" explanation given the train data.

Acknowledgments

I am grateful to the editor and the anonymous reviewers for their useful comments.

Appendix A

In this appendix, we first present the combinatorial formula used to evaluate the numbers of partition of a set and second provide the proof of the theorems introduced in the paper.

A.1 Partition numbers

The number of ways a set of n elements can be partitioned into nonempty subsets is called a Bell number and denoted B_n .

The number of ways of partitioning a set of n elements into k nonempty subsets is called a Stirling number of the second kind and denoted $S(n, k)$.

The Bell numbers can be defined by the sum

$$B_n = \sum_{k=1}^n S(n, k) .$$

Let $B(n, k)$ be the number of partition of a set of n elements into at most k parts. This number, that we choose to call a generalized Bell number, can be defined by the sum

$$B(n, k) = \sum_{i=1}^k S(n, i) .$$

Theorem 1: A SGM model distributed according to the three-stage prior is Bayes optimal for a given set of categorical values if the value of the following criterion is minimal on the set of all SGM models:

$$\log(I) + \log(B(I, K)) + \sum_{k=1}^K \log(C_{n_k+J-1}^{J-1}) + \sum_{k=1}^K \log(n_k! / n_{k,1}! n_{k,2}! \dots n_{k,J}!).$$

Proof:

The prior probability of a grouping model M can be defined by the prior probability of the parameters of the model.

Let us introduce some notations:

- $p(K)$: prior probability of the number of groups K ,
- $p(\{k(i)\})$: prior probability of a partition (defined by $\{k(i)\}$) of the categorical values into K groups,
- $p(\{n_{kj}\})$: prior probability of the set of parameters $\{n_{11}, \dots, n_{kj}, \dots, n_{KJ}\}$,
- $p(\{n_{kj}\}_k)$: prior probability of the set of parameters $\{n_{k1}, \dots, n_{kJ}\}$.

The objective is to find the grouping model M that maximizes the probability $p(M | D)$ for a given train data set D . Using Bayes formula and since the probability $p(D)$ is constant under varying the model, this is equivalent to maximize $p(M)p(D|M)$.

Let us first focus on the prior probability $p(M)$ of the model. We have

$$\begin{aligned} p(M) &= p(K, \{k(i)\}, \{n_{kj}\}) \\ &= p(K) p(\{k(i)\} | K) p(\{n_{kj}\} | K, \{k(i)\}). \end{aligned}$$

The first hypothesis of the three-stage prior is that the number of groups is uniformly distributed between 1 and I . Thus we get

$$p(K) = \frac{1}{I}.$$

The second hypothesis is that all the partition of the categorical values into at most K groups are equiprobable for a given K . Computing the probability of one set of groups turns into the combinatorial evaluation of the number of possible group sets. This number is equal to the generalized Bell number $B(I, K)$. Thus we obtain

$$p(\{k(i)\} | K) = \frac{1}{B(I, K)}.$$

The last term to evaluate can be rewritten as a product using the hypothesis of independence of the distributions of the class values between the groups. We have

$$\begin{aligned} p(\{n_{kj}\} | K, \{k(i)\}) &= p(\{n_{kj}\}_1, \{n_{kj}\}_2, \dots, \{n_{kj}\}_K | K, \{k(i)\}) \\ &= \prod_{k=1}^K p(\{n_{kj}\}_k | K, \{k(i)\}) \\ &= \prod_{k=1}^K p(\{n_{kj}\}_k | n_k). \end{aligned}$$

The frequencies per group n_k derive from the frequencies per categorical values n_i for a given partition of the values into groups.

For a given group k with size n_k , all the distributions of the class values are equiprobable. Computing the probability of one distribution is a combinatorial problem, which solution is

$$p(\{n_{kj}\}_k | n_k) = \frac{1}{C_{n_k+J-1}^{J-1}}.$$

Thus,

$$p(\{n_{kj}\} | K, \{k(i)\}) = \prod_{k=1}^K \frac{1}{C_{n_k+J-1}^{J-1}}.$$

The prior probability of the model is then

$$p(M) = \frac{1}{I} \frac{1}{B(I, K)} \prod_{k=1}^K \frac{1}{C_{n_k+J-1}^{J-1}}.$$

Let us now evaluate the probability of getting the train data set D for a given model M . We first divide the data set D into K subsets D_k of size n_k corresponding to the K groups. Using again the independence assumption between the groups, we obtain

$$\begin{aligned} p(D | M) &= p(D | K, \{k(i)\}, \{n_{kj}\}) \\ &= p(D_1, D_2, \dots, D_K | K, \{k(i)\}, \{n_{kj}\}) \\ &= \prod_{k=1}^K p(D_k | K, \{k(i)\}, \{n_{kj}\}) \\ &= \prod_{k=1}^K \frac{1}{(n_k! / n_{k,1}! n_{k,2}! \dots n_{k,J}!)}, \end{aligned}$$

as evaluating the probability of a subset D_k under uniform prior turns out to be a multinomial problem.

Taking the negative log of the probabilities, the maximization problem turns into the minimization of the claimed criterion

$$\log(I) + \log(B(I, K)) + \sum_{k=1}^K \log(C_{n_k+J-1}^{J-1}) + \sum_{k=1}^K \log(n_k! / n_{k,1}! n_{k,2}! \dots n_{k,J}!).$$

Theorem 2: In a Bayes optimal SGM model distributed according to the three-stage prior, two pure categorical values having the same class are necessary in the same group.

Proof:

Let a and b be the two pure values related to the same class (indexed as class 1 for convenience reasons). Let us assume that, contrary to the claim, the two values are separated into two groups $A1$ and $B1$.

We construct two new groups $A0$ and $B2$ by moving the pure value a from $A1$ to $B1$. The cost variation $\Delta Cost1$ of the grouping is

$$\begin{aligned} \Delta Cost1 &= \Delta PartitionCost1 \\ &+ \log((n_{A0} + J - 1)! / n_{A0}!(J - 1)!) + \log(n_{A0}! / n_{A0,1}! n_{A0,2}! \dots n_{A0,J}!) \\ &+ \log((n_{B2} + J - 1)! / n_{B2}!(J - 1)!) + \log(n_{B2}! / n_{B2,1}! n_{B2,2}! \dots n_{B2,J}!) \\ &- \log((n_{A1} + J - 1)! / n_{A1}!(J - 1)!) - \log(n_{A1}! / n_{A1,1}! n_{A1,2}! \dots n_{A1,J}!) \\ &- \log((n_{B1} + J - 1)! / n_{B1}!(J - 1)!) - \log(n_{B1}! / n_{B1,1}! n_{B1,2}! \dots n_{B1,J}!). \end{aligned}$$

If the $A1$ group contains only the a value, moving a from $A1$ to $B1$ results in a decreased number of groups, with a related variation of partition cost

$$\Delta PartitionCost1 = \log(B(I, K - 1)) - \log(B(I, K)),$$

which is negative. In the opposite case, the number of groups remains the same and the resulting variation of partition cost is zero.

The frequencies are the same for each class except for class 1, thus

$$\begin{aligned} \Delta Cost1 - \Delta PartitionCost1 &= \log\left(\frac{(n_{A1} - n_a + J - 1)!}{(n_{A1} + J - 1)!}\right) \\ &\quad + \log\left(\frac{(n_{B1} + n_a + J - 1)!}{(n_{B1} + J - 1)!}\right) \\ &\quad + \log\left(\frac{n_{A1,1}!}{(n_{A1,1} - n_{a,1})!}\right) \\ &\quad + \log\left(\frac{n_{B1,1}!}{(n_{B1,1} + n_{a,1})!}\right), \\ \Delta Cost1 - \Delta PartitionCost1 &= \log\left(\frac{\prod_{n=0}^{n_a-1} (n_{A1,1} - n)}{(n_{A1} - n + J - 1)}\right) \\ &\quad - \log\left(\frac{\prod_{n=1}^{n_a} (n_{B1,1} + n)}{(n_{B1} + n + J - 1)}\right). \end{aligned}$$

Similarly, we construct two groups $A2$ and $B0$ by moving pure value b from $B1$ to $A1$. This time, the cost variation $\Delta Cost2$ of the grouping is

$$\begin{aligned} \Delta Cost2 - \Delta PartitionCost2 &= \log\left(\frac{\prod_{n=0}^{n_b-1} (n_{B1,1} - n)}{(n_{B1} - n + J - 1)}\right) \\ &\quad - \log\left(\frac{\prod_{n=1}^{n_b} (n_{A1,1} + n)}{(n_{A1} + n + J - 1)}\right). \end{aligned}$$

Let us assume that

$$n_{A1,1}/(n_{A1} + J - 1) \leq n_{B1,1}/(n_{B1} + J - 1).$$

Using the property

$$0 \leq z \leq x < y \Rightarrow (x - z)/(y - z) \leq x/y \leq (x + z)/(y + z),$$

we obtain

$$\begin{aligned} \prod_{n=0}^{n_a-1} (n_{A1,1} - n)/(n_{A1} - n + J - 1) &\leq (n_{A1,1}/n_{A1} + J - 1)^{n_a} \\ &\leq (n_{B1,1}/n_{B1} + J - 1)^{n_a} \leq \prod_{n=1}^{n_a} (n_{B1,1} + n)/(n_{B1} + n + J - 1). \end{aligned}$$

Thus, we get $\Delta Cost1 - \Delta PartitionCost1 \leq 0$.

On the opposite, let us assume that

$$n_{A1,1}/(n_{A1} + J - 1) \geq n_{B1,1}/(n_{B1} + J - 1).$$

This time, we obtain $\Delta Cost2 - \Delta PartitionCost2 \leq 0$.

Since the variations of partition costs are always non-negative, at least one of the two cost variations $\Delta Cost1$ or $\Delta Cost2$ is negative and the initial grouping could not be optimal. As this is contradictory with the initial assumption, the claim follows.

Remark:

If the partition costs are evaluated using the Stirling numbers of the second kind instead of the generalized Bell numbers, this theorem is no longer true, since decreasing the number of groups can result in an increase of the partition cost (for example, $S(I, I-1) = I(I-1)/2$ and $S(I, I) = 1$).

Theorem 3: In a SGM model distributed according to the three-stage prior and in the case of two classes, the Bayes optimal grouping model consists of a single group when each instance has a different categorical value.

Proof:

Since each instance has a different categorical value, all the categorical values are pure values associated with one among the J class values. According to Theorem 2, the values having the same class values are necessary in the same group. The optimal grouping contains at most J groups.

Let A and B be two groups and $A \cup B$ the group obtained by merging A and B .

Let n_A , n_B and $n_{A \cup B}$ be the frequencies of these groups, and let $n_{A,j}$, $n_{B,j}$ and $n_{A \cup B,j}$ be the frequencies per class value in these groups.

When the two groups are merged, the number of groups decreases from K to $K-1$ with the variation of partition cost $\Delta PartitionCost = \log(B(I, K-1)) - \log(B(I, K))$.

The total variation of the grouping cost is

$$\begin{aligned} \Delta Cost &= \Delta PartitionCost \\ &+ \left(\log \left(C_{n_{A \cup B} + J - 1}^{J-1} \right) + \log \left(n_{A \cup B}! / n_{A \cup B,1}! n_{A \cup B,2}! \dots n_{A \cup B,J}! \right) \right) \\ &- \left(\log \left(C_{n_A + J - 1}^{J-1} \right) + \log \left(n_A! / n_{A,1}! n_{A,2}! \dots n_{A,J}! \right) \right) \\ &- \left(\log \left(C_{n_B + J - 1}^{J-1} \right) + \log \left(n_B! / n_{B,1}! n_{B,2}! \dots n_{B,J}! \right) \right), \\ \Delta Cost &= \Delta PartitionCost \\ &+ \log \left((n_{A \cup B} + J - 1)! (J - 1)! / (n_A + J - 1)! (n_B + J - 1)! \right) \\ &- \sum_{j=1}^J \log \left(C_{n_{A \cup B,j}}^{n_{A,j}} \right). \end{aligned}$$

Since each class is fully contained either in group A or B , we obtain

$$\Delta Cost = \Delta PartitionCost + \log \left((n_{A \cup B} + J - 1)! (J - 1)! / (n_A + J - 1)! (n_B + J - 1)! \right).$$

We are in the case of 2 class values and thus have $J = 2$, $n_{A \cup B} = n$, $K = 2$.

$$\begin{aligned} \Delta Cost &= \log(B(n, 1)) - \log(B(n, 2)) + \log \left((n+1)! / (n_A+1)! (n_B+1)! \right) \\ &= -\log(2^{n-1}) + \log \left(C_{n+2}^{n_A+1} \right) - \log(n+2). \end{aligned}$$

Since we have $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k \leq 2^{n-1}$ for $n > 1$ and $k > 1$ (with a strict inequality when $n > 2$), we finally obtain

$$\begin{aligned} \Delta Cost &< -\log(2^{n-1}) + \log(2^{n+1}) - \log(n+2) \\ &< \log(4/(n+2)) \\ &< 0. \end{aligned}$$

The claim follows.

Remark:

If the partition costs are evaluated using the Stirling numbers of the second kind instead of the generalized Bell numbers, this theorem is no longer true. In particular, when the class values are equi-distributed, the grouping cost in the case of a single group ($\log(n) + \log(n+1) + \log(C_n^{n/2})$) is higher than the grouping cost in the case of one group per categorical value ($\log(n) + n \log(2)$).

References

M. Asseraf. Metric on decision trees and optimal partition problem. *International Conference on Human System Learning, Proceedings of CAPS'3*, Paris, 2000.

- N. C. Berckman. Value grouping for binary decision trees. Technical Report, Computer Science Department – University of Massachusetts, 1995.
- J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley, New York, 1994.
- C. L. Blake and C. J. Merz. UCI Repository of machine learning databases Web URL <http://www.ics.uci.edu/~mllearn/MLRepository.html>. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
- M. Boullé. Khiops: a Statistical Discretization Method of Continuous Attributes. *Machine Learning*, 55(1):53-69, 2004a.
- M. Boullé. A robust method for partitioning the values of categorical attributes. *Revue des Nouvelles Technologies de l'Information, Extraction et gestion des connaissances (EGC'2004)*, RNTI-E-2, volume II: 173-182, 2004b.
- M. Boullé. A Bayesian Approach for Supervised Discretization. *Data Mining V*, Eds Zanasi, Ebecken, Brebbia, WIT Press, pp 199-208, 2004c.
- L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone. *Classification and Regression Trees*. California: Wadsworth International, 1984.
- B. Cestnik, I. Kononenko and I. Bratko. ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In Bratko and Lavrac (Eds.), *Progress in Machine Learning*. Wilmslow, UK: Sigma Press, 1987.
- P. A. Chou. Optimal Partitioning for Classification and Regression Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):340-354, 1991.
- T. G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Methods. *Neural Computation*, 10(7), 1998.
- J. Dougherty, R. Kohavi and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the Twelfth International Conference on Machine Learning*. Los Altos, CA: Morgan Kaufmann, pp 194-202, 1995.
- T. Fulton, S. Kasif and S. Salzberg. Efficient algorithms for finding multi-way splits for decision trees. In *Proc. Thirteenth International Joint Conference on Artificial Intelligence*, San Francisco, CA: Morgan Kaufmann, pp 244-255, 1995.
- D. J. Hand and K. Yu. Idiot Bayes ? not so stupid after all? *International Statistical Review*, 69:385-398, 2001.
- C. N. Hsu, H. J. Huang and T. T Wong. Implications of the Dirichlet Assumption for Discretization of Continuous Variables in Naive Bayesian Classifiers. *Machine Learning*, 53(3):235-263, 2003.
- G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2):119-127, 1980.
- R. Kass and A. Raftery. Bayes factors. In *Journal of the American Statistical Association*, 90: 773-795, 1995.
- R. Kerber. Chimerge discretization of numeric attributes. *Proceedings of the 10th International Conference on Artificial Intelligence*, pp 123-128, 1991.
- S. Kullback. *Information Theory and Statistics*. New York: Wiley, (1959); republished by Dover, 1968.

- P. Langley, W. Iba and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the 10th national conference on Artificial Intelligence*, AAAI Press, pp 223-228, 1992.
- Y. Lechevallier. Recherche d'une partition optimale sous contrainte d'ordre total. Technical report N°1247. INRIA, 1990.
- D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999.
- J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- G. Ritschard, D. A. Zighed and N. Nicoloyannis. Maximisation de l'association par regroupement de lignes ou de colonnes d'un tableau croisé. *Mathématiques et Sciences Humaines*, n°154-155:81-98, 2001.
- G. Ritschard. Partition BIC optimale de l'espace des prédicteurs. *Revue des Nouvelles Technologies de l'Information*, 1:99-110, 2003.
- G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461-464, 1978.
- SPSS Inc. *AnswerTree 3.0 User's Guide*. Chicago: SPSS Inc, 2001.
- Y. Yang and G. Webb. On why discretization works for naïve-Bayes classifiers. *Proceedings of the 16th Australian Joint Conference on Artificial Intelligence (AI)*, 2003.
- D. A. Zighed and R. Rakotomalala. *Graphes d'induction*. Hermes Science Publications, pp 327-359, 2000.

Large Margin Methods for Structured and Interdependent Output Variables

Ioannis Tsochantaridis

Google, Inc.

Mountain View, CA 94043, USA

IOANNIS@GOOGLE.COM

Thorsten Joachims

Department of Computer Science

Cornell University

Ithaca, NY 14853, USA

TJ@CS.CORNELL.EDU

Thomas Hofmann

Darmstadt University of Technology

Fraunhofer IPSI

Darmstadt, Germany

HOFMANN@INT.TU-DARMSTADT.DE

Yasemin Altun

Toyota Technological Institute

Chicago, IL 60637, USA

ALTUN@TTI-C.ORG

Editor: Yoram Singer

Abstract

Learning general functional dependencies between arbitrary input and output spaces is one of the key challenges in computational intelligence. While recent progress in machine learning has mainly focused on designing flexible and powerful input representations, this paper addresses the complementary issue of designing classification algorithms that can deal with more complex outputs, such as trees, sequences, or sets. More generally, we consider problems involving multiple dependent output variables, structured output spaces, and classification problems with class attributes. In order to accomplish this, we propose to appropriately generalize the well-known notion of a separation margin and derive a corresponding maximum-margin formulation. While this leads to a quadratic program with a potentially prohibitive, i.e. exponential, number of constraints, we present a cutting plane algorithm that solves the optimization problem in polynomial time for a large class of problems. The proposed method has important applications in areas such as computational biology, natural language processing, information retrieval/extraction, and optical character recognition. Experiments from various domains involving different types of output spaces emphasize the breadth and generality of our approach.

1. Introduction

This paper deals with the general problem of learning a mapping from input vectors or patterns $\mathbf{x} \in \mathcal{X}$ to discrete response variables $\mathbf{y} \in \mathcal{Y}$, based on a training sample of input-output pairs $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y}$ drawn from some fixed but unknown probability distribution. Unlike multiclass classification, where the output space consists of an arbitrary finite set of labels or class identifiers, $\mathcal{Y} = \{1, \dots, K\}$, or regression, where $\mathcal{Y} = \mathbb{R}$ and the response variable is a scalar, we consider the case where elements of \mathcal{Y} are *structured objects* such as sequences, strings, trees,

lattices, or graphs. Such problems arise in a variety of applications, ranging from multilabel classification and classification with class taxonomies, to label sequence learning, sequence alignment learning, and supervised grammar learning, to name just a few. More generally, these problems fall into two generic cases: first, problems where classes themselves can be characterized by certain class-specific attributes and learning should occur across classes as much as across patterns; second, problems where \mathbf{y} represents a macro-label, i.e. describes a *configuration* over components or state variables $\mathbf{y} = (y^1, \dots, y^T)$, with possible dependencies between these state variables.

We approach these problems by generalizing large margin methods, more specifically multiclass support vector machines (SVMs) (Weston and Watkins, 1998; Crammer and Singer, 2001), to the broader problem of learning structured responses. The naive approach of treating each structure as a separate class is often intractable, since it leads to a multiclass problem with a very large number of classes. We overcome this problem by specifying discriminant functions that exploit the structure and dependencies within \mathcal{Y} . In that respect our approach follows the work of Collins (2002) on perceptron learning with a similar class of discriminant functions. However, the maximum-margin algorithm we propose has advantages in terms of accuracy and tunability to specific loss functions. A maximum-margin algorithm has also been proposed by Collins and Duffy (2002a) in the context of natural language processing. However, it depends on the size of the output space, therefore it requires some external process to enumerate a small number of candidate outputs \mathbf{y} for a given input \mathbf{x} . The same is true also for other ranking algorithms (Cohen et al., 1999; Herbrich et al., 2000; Schapire and Singer, 2000; Crammer and Singer, 2002; Joachims, 2002). In contrast, we have proposed an efficient algorithm (Hofmann et al., 2002; Altun et al., 2003; Joachims, 2003) even in the case of very large output spaces, that takes advantage of the sparseness of the maximum-margin solution.

A different maximum-margin algorithm that can deal with very large output sets, maximum margin Markov networks, has been independently proposed by Taskar et al. (2004a). The structure of the output is modeled by a Markov network, and by employing a probabilistic interpretation of the dual formulation of the problem, Taskar et al. (2004a) propose a reparameterization of the problem, that leads to an efficient algorithm, as well as generalization bounds that do not depend on the size of the output space. The proposed reparameterization, however, assumes that the loss function can be decomposed in the the same fashion as the feature map, thus does not support arbitrary loss functions that may be appropriate for specific applications.

On the surface our approach is related to the kernel dependency estimation approach described in Weston et al. (2003). There, however, separate kernel functions are defined for the input and output space, with the idea to encode a priori knowledge about the similarity or loss function in output space. In particular, this assumes that the loss is input dependent and known beforehand. More specifically, in Weston et al. (2003) a kernel PCA is used in the feature space defined over \mathbf{y} to reduce the problem to a (small) number of independent regression problems. The latter corresponds to an unsupervised embedding (followed by dimension reduction) performed in the output space and no information about the patterns \mathbf{x} is utilized in defining this low-dimensional representation. In contrast, the key idea in our approach is not primarily to define more complex functions, but to deal with more complex output spaces by extracting combined features over inputs and outputs.

For a large class of structured models, we propose a novel SVM algorithm that allows us to learn mappings involving complex structures in polynomial time despite an exponential (or infinite) number of possible output values. In addition to respective theoretical results, we empirically evaluate our approach for a number of specific problem instantiations: classification with class tax-

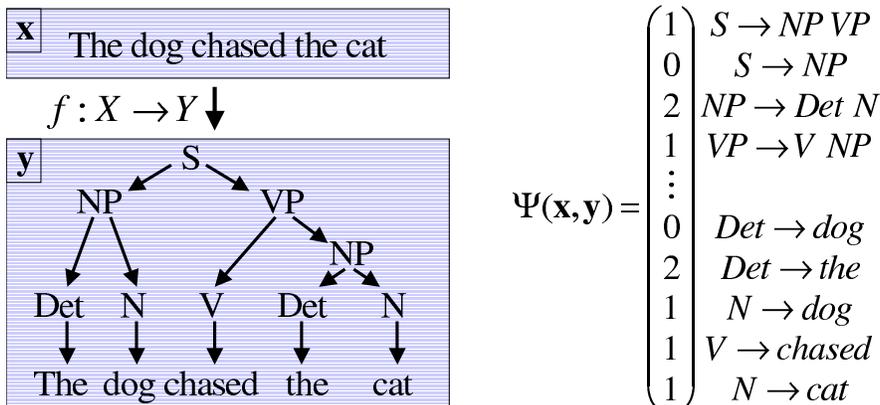


Figure 1: Illustration of natural language parsing model.

onomies, label sequence learning, sequence alignment, and natural language parsing. This paper extends Tsochantaridis et al. (2004) with additional theoretical and empirical results.

The rest of the paper is organized as follows: Section 2 presents the general framework of large margin learning over structured output spaces using representations of input-output pairs via joint feature maps. Section 3 describes and analyzes a generic algorithm for solving the resulting optimization problems. Sections 4 and 5 discuss numerous important special cases and experimental results, respectively.

2. Large Margin Learning with Joint Feature Maps

We are interested in the general problem of learning functions $f : X \rightarrow \mathcal{Y}$ between input spaces X and arbitrary discrete output spaces \mathcal{Y} based on a training sample of input-output pairs. As an illustrating example, which we will continue to use as a prototypical application in the sequel, consider the case of natural language parsing, where the function f maps a given sentence \mathbf{x} to a parse tree \mathbf{y} . This is depicted graphically in Figure 1.

The approach we pursue is to learn a *discriminant function* $F : X \times \mathcal{Y} \rightarrow \mathbb{R}$ over input-output pairs from which we can derive a prediction by maximizing F over the response variable for a specific given input \mathbf{x} . Hence, the general form of our hypotheses f is

$$f(\mathbf{x}; \mathbf{w}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}), \tag{1}$$

where \mathbf{w} denotes a parameter vector. It might be useful to think of F as a compatibility function that measures how compatible pairs (\mathbf{x}, \mathbf{y}) are, or, alternatively, $-F$ can be thought of as a \mathbf{w} -parameterized family of cost functions, which we try to design in such a way that the minimum of $F(\mathbf{x}, \cdot; \mathbf{w})$ is at the desired output \mathbf{y} for inputs \mathbf{x} of interest.

Throughout this paper, we assume F to be linear in some *combined feature representation* of inputs and outputs $\Psi(\mathbf{x}, \mathbf{y})$, i.e.

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle. \tag{2}$$

The specific form of Ψ depends on the nature of the problem and special cases will be discussed subsequently. However, whenever possible we will develop learning algorithms and theoretical

results for the general case. Since we want to exploit the advantages of kernel-based method, we will pay special attention to cases where the inner product in the joint representation can be efficiently computed via a joint kernel function $J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', \mathbf{y}') \rangle$.

Using again natural language parsing as an illustrative example, we can chose F such that we get a model that is isomorphic to a probabilistic context free grammar (PCFG) (cf. Manning and Schuetze, 1999). Each node in a parse tree \mathbf{y} for a sentence \mathbf{x} corresponds to grammar rule g_j , which in turn has a score w_j . All valid parse trees \mathbf{y} (i.e. trees with a designated start symbol S as the root and the words in the sentence \mathbf{x} as the leaves) for a sentence \mathbf{x} are scored by the sum of the w_j of their nodes. This score can thus be written in the form of Equation (2), with $\Psi(\mathbf{x}, \mathbf{y})$ denoting a histogram vector of counts (how often each grammar rule g_j occurs in the tree \mathbf{y}). $f(\mathbf{x}; \mathbf{w})$ can be efficiently computed by finding the structure $\mathbf{y} \in Y$ that maximizes $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ via the CKY algorithm (Younger, 1967; Manning and Schuetze, 1999).

2.1 Loss Functions and Risk Minimization

The standard zero-one loss function typically used in classification is not appropriate for most kinds of structured responses. For example, in natural language parsing, a parse tree that is almost correct and differs from the correct parse in only one or a few nodes should be treated differently from a parse tree that is completely different. Typically, the correctness of a predicted parse tree is measured by its F_1 score (see e.g. Johnson, 1998), the harmonic mean of precision and recall as calculated based on the overlap of nodes between the trees.

In order to quantify the accuracy of a prediction, we will consider learning with arbitrary loss functions $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$. Here $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ quantifies the loss associated with a prediction $\hat{\mathbf{y}}$, if the true output value is \mathbf{y} . It is usually sufficient to restrict attention to zero diagonal loss functions with $\Delta(\mathbf{y}, \mathbf{y}) = 0$ and for which furthermore $\Delta(\mathbf{y}, \mathbf{y}') > 0$ for $\mathbf{y} \neq \mathbf{y}'$.¹ Moreover, we assume the loss is bounded for every given target value \mathbf{y}^* , i.e. $\max_{\mathbf{y}} \{\Delta(\mathbf{y}^*, \mathbf{y})\}$ exists.

We investigate a supervised learning scenario, where input-output pairs (\mathbf{x}, \mathbf{y}) are generated according to some fixed distribution $P(\mathbf{x}, \mathbf{y})$ and the goal is to find a function f in a given hypothesis class such that the risk,

$$\mathcal{R}_P^\Delta(f) = \int_{\mathcal{X} \times \mathcal{Y}} \Delta(\mathbf{y}, f(\mathbf{x})) dP(\mathbf{x}, \mathbf{y}),$$

is minimized. Of course, P is unknown and following the supervised learning paradigm, we assume that a finite training set of pairs $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, \dots, n\}$ generated i.i.d. according to P is given. The performance of a function f on the training sample S is described by the empirical risk,

$$\mathcal{R}_S^\Delta(f) = \frac{1}{n} \sum_{i=1}^n \Delta(\mathbf{y}_i, f(\mathbf{x}_i)),$$

which is simply the expected loss under the empirical distribution induced by S . For \mathbf{w} -parameterized hypothesis classes, we will also write $\mathcal{R}_P^\Delta(\mathbf{w}) \equiv \mathcal{R}_P^\Delta(f(\cdot; \mathbf{w}))$ and similarly for the empirical risk.

1. Cases where $\Delta(\mathbf{y}, \mathbf{y}') = 0$ for $\mathbf{y} \neq \mathbf{y}'$ can be dealt with, but lead to additional technical overhead, which we chose to avoid for the sake of clarity.

2.2 Margin Maximization

We consider various scenarios for the generalization of support vector machine learning over structured outputs. We start with the simple case of hard-margin SVMs, followed by soft-margin SVMs, and finally we propose two approaches for the case of loss-sensitive SVMs, which is the most general case and subsumes the former ones.

2.2.1 SEPARABLE CASE

First, we consider the case where there exists a function f parameterized by \mathbf{w} such that the empirical risk is zero. The condition of zero training error can then be compactly written as a set of nonlinear constraints

$$\forall i \in \{1, \dots, n\}: \quad \max_{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i} \{\langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle\} \leq \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i) \rangle. \quad (3)$$

Notice that this holds independently of the loss functions, since we have assumed that $\Delta(\mathbf{y}, \mathbf{y}) = 0$ and $\Delta(\mathbf{y}, \mathbf{y}') > 0$ for $\mathbf{y} \neq \mathbf{y}'$.

Every one of the nonlinear inequalities in Equation (3) can be equivalently replaced by $|\mathcal{Y}| - 1$ linear inequalities, resulting in a total of $n|\mathcal{Y}| - n$ linear constraints,

$$\forall i \in \{1, \dots, n\}, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i: \quad \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y}) \rangle \geq 0. \quad (4)$$

As we will often encounter terms involving feature vector differences of the type appearing in Equation (4), we define $\delta\Psi_i(\mathbf{y}) \equiv \Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y})$ so that the constraints can be more compactly written as $\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq 0$.

If the set of inequalities in Equation (4) is feasible, there will typically be more than one solution \mathbf{w}^* . To specify a unique solution, we propose to select the \mathbf{w} for which the separation margin γ , i.e. the minimal differences between the score of the correct label \mathbf{y}_i and the closest runner-up $\hat{\mathbf{y}}(\mathbf{w}) = \operatorname{argmax}_{\mathbf{y} \neq \mathbf{y}_i} \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$, is maximal. This generalizes the maximum-margin principle employed in support vector machines (Vapnik, 1998) to the more general case considered in this paper. Restricting the L_2 norm of \mathbf{w} to make the problem well-posed leads to the following optimization problem:

$$\begin{aligned} & \max_{\gamma, \mathbf{w}: \|\mathbf{w}\|=1} \gamma \\ & \text{s.t. } \forall i \in \{1, \dots, n\}, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i: \quad \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq \gamma. \end{aligned}$$

This problem can be equivalently expressed as a convex quadratic program in standard form

$$\text{SVM}_0: \quad \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 \quad (5)$$

$$\text{s.t. } \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i: \quad \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq 1. \quad (6)$$

2.2.2 SOFT-MARGIN MAXIMIZATION

To allow errors in the training set, we introduce slack variables and propose to optimize a soft-margin criterion. As in the case of multiclass SVMs, there are at least two ways of introducing slack variables. One may introduce a single slack variable ξ_i for violations of the *nonlinear* constraints

(i.e. every instance \mathbf{x}_i) (Crammer and Singer, 2001) or one may penalize margin violations for every *linear* constraint (i.e. every instance \mathbf{x}_i and output $\mathbf{y} \neq \mathbf{y}_i$) (Weston and Watkins, 1998; Har-Peled et al., 2002). Since the former will result in a (tighter) upper bound on the empirical risk (cf. Proposition 1) and offers some advantages in the proposed optimization scheme (cf. Section 3), we have focused on this formulation. Adding a penalty term that is linear in the slack variables to the objective, results in the quadratic program

$$\begin{aligned} \text{SVM}_1 : \quad & \min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ & \text{s.t. } \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle \geq 1 - \xi_i, \xi_i \geq 0. \end{aligned} \quad (7)$$

Alternatively, we can also penalize margin violations by a quadratic term leading to the following optimization problem:

$$\begin{aligned} \text{SVM}_2 : \quad & \min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2n} \sum_{i=1}^n \xi_i^2 \\ & \text{s.t. } \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle \geq 1 - \xi_i. \end{aligned}$$

In both cases, $C > 0$ is a constant that controls the trade-off between training error minimization and margin maximization.

2.2.3 GENERAL LOSS FUNCTIONS: SLACK RE-SCALING

The first approach we propose for the case of arbitrary loss functions, is to re-scale the slack variables according to the loss incurred in each of the linear constraints. Intuitively, violating a margin constraint involving a $\mathbf{y} \neq \mathbf{y}_i$ with high loss $\Delta(\mathbf{y}_i, \mathbf{y})$ should be penalized more severely than a violation involving an output value with smaller loss. This can be accomplished by multiplying the margin violation by the loss, or equivalently, by scaling the slack variable with the inverse loss, which yields

$$\begin{aligned} \text{SVM}_1^{\Delta_s} : \quad & \min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^n \xi_i \\ & \text{s.t. } \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{y}_i, \mathbf{y})}. \end{aligned}$$

A justification for this formulation is given by the subsequent proposition.

Proposition 1 *Denote by $\xi^*(\mathbf{w})$ the optimal solution of the slack variables in $\text{SVM}_1^{\Delta_s}$ for a given weight vector \mathbf{w} . Then $\frac{1}{n} \sum_{i=1}^n \xi_i^*$ is an upper bound on the empirical risk $\mathcal{R}_S^{\Delta}(\mathbf{w})$.*

Proof *Notice first that $\xi_i^* = \max\{0, \max_{\mathbf{y} \neq \mathbf{y}_i} \{\Delta(\mathbf{y}_i, \mathbf{y}) (1 - \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle)\}\}$.*

Case 1: If $f(\mathbf{x}_i; \mathbf{w}) = \mathbf{y}_i$, then $\xi_i^ \geq 0 = \Delta(\mathbf{y}_i, f(\mathbf{x}_i; \mathbf{w}))$ and the loss is trivially upper bounded.*

Case 2: If $\hat{\mathbf{y}} \equiv f(\mathbf{x}_i; \mathbf{w}) \neq \mathbf{y}_i$, then $\langle \mathbf{w}, \delta \Psi_i(\hat{\mathbf{y}}) \rangle \leq 0$ and thus $\frac{\xi_i^}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \geq 1$ which is equivalent to $\xi_i^* \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}})$.*

Since the bound holds for every training instance, it also holds for the average. ■

The optimization problem $\text{SVM}_2^{\Delta_s}$ can be derived analogously, where $\Delta(\mathbf{y}_i, \mathbf{y})$ is replaced by $\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}$ in order to obtain an upper bound on the empirical risk.

2.2.4 GENERAL LOSS FUNCTIONS: MARGIN RE-SCALING

In addition to this *slack re-scaling* approach, a second way to include loss functions is to re-scale the margin as proposed by Taskar et al. (2004a) for the special case of the Hamming loss. It is straightforward to generalize this method to general loss functions. The margin constraints in this setting take the following form:

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} : \quad \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i. \quad (8)$$

The set of constraints in Equation (8) combined with the objective in Equation (7) yield an optimization problem $\text{SVM}_1^{\Delta m}$ which also results in an upper bound on $\mathcal{R}_S^\Delta(\mathbf{w}^*)$.

Proposition 2 Denote by $\xi^*(\mathbf{w})$ the optimal solution of the slack variables in $\text{SVM}_1^{\Delta m}$ for a given weight vector \mathbf{w} . Then $\frac{1}{n} \sum_{i=1}^n \xi_i^*$ is an upper bound on the empirical risk $\mathcal{R}_S^\Delta(\mathbf{w})$.

Proof The essential observation is that $\xi_i^* = \max\{0, \max_{\mathbf{y}}\{\Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle\}\}$ which is guaranteed to upper bound $\Delta(\mathbf{y}_i, \mathbf{y})$ for \mathbf{y} such that $\langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle \leq 0$. ■

The optimization problem $\text{SVM}_2^{\Delta m}$ can be derived analogously, where $\Delta(\mathbf{y}_i, \mathbf{y})$ is replaced by $\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}$.

2.2.5 GENERAL LOSS FUNCTIONS: DISCUSSION

Let us discuss some of the advantages and disadvantages of the two formulations presented. An appealing property of the slack re-scaling approach is its scaling invariance.

Proposition 3 Suppose $\Delta' \equiv \eta \Delta$ with $\eta > 0$, i.e. Δ' is a scaled version of the original loss Δ . Then by re-scaling $C' = C/\eta$, the optimization problems $\text{SVM}_1^{\Delta s}(C)$ and $\text{SVM}_1^{\Delta' s}(C')$ are equivalent as far as \mathbf{w} is concerned. In particular the optimal weight vector \mathbf{w}^* is the same in both cases.

Proof First note that each \mathbf{w} is feasible for $\text{SVM}_1^{\Delta s}$ and $\text{SVM}_1^{\Delta' s}$ in the sense that we can find slack variables such that all the constraints are satisfied. In fact we can chose them optimally and define $H(\mathbf{w}) \equiv \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_i \xi_i^*(\mathbf{w})$ and $H'(\mathbf{w}) \equiv \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C'}{n} \sum_i \xi_i^{*'}(\mathbf{w})$, where ξ_i^* and $\xi_i^{*'}$ refer to the optimal slacks in $\text{SVM}_1^{\Delta s}$ and $\text{SVM}_1^{\Delta' s}$, respectively, for given \mathbf{w} . It is easy to see that they are given by

$$\xi_i^* = \max\{0, \max_{\mathbf{y} \neq \mathbf{y}_i} \{\Delta(\mathbf{y}_i, \mathbf{y}) (1 - \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle)\}\}$$

and

$$\xi_i^{*' } = \max\{0, \max_{\mathbf{y} \neq \mathbf{y}_i} \{\eta \Delta(\mathbf{y}_i, \mathbf{y}) (1 - \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle)\}\},$$

respectively. Pulling η out of the max, one gets that $\xi_i^{*' } = \eta \xi_i^*$ and thus $\sum_i \xi_i^* = C \eta \sum_i \xi_i^{*' } = C' \sum_i \xi_i^{*' }$. From that it follows immediately that $H = H'$. ■

In contrast, the margin re-scaling formulation is not invariant under scaling of the loss function. One needs, for example, to re-scale the feature map Ψ by a corresponding scale factor as well. This seems to indicate that one has to calibrate the scaling of the loss and the scaling of the feature map

more carefully in the $SVM_1^{\Delta m}$ formulation. The $SVM_1^{\Delta s}$ formulation on the other hand, represents the loss scale explicitly in terms of the constant C .

A second disadvantage of the margin scaling approach is that it potentially gives significant weight to output values $\mathbf{y} \in \mathcal{Y}$ that are not even close to being confusable with the target values \mathbf{y}_i , because every increase in the loss increases the required margin. If one interprets $F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) - F(\mathbf{x}_i, \mathbf{y}; \mathbf{w})$ as a log odds ratio of an exponential family model (Smola and Hofmann, 2003), then the margin constraints may be dominated by incorrect values \mathbf{y} that are exponentially less likely than the target value. To be more precise, notice that in the $SVM_1^{\Delta s}$ formulation, the penalty part only depends on \mathbf{y} for which $\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \leq 1$. These are output values \mathbf{y} that all receive a relatively “high” (i.e. 1-close to the optimum) value of $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$. However, in $SVM_1^{\Delta m}$, ξ_i^* has to majorize $\Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle$ for all \mathbf{y} . This means ξ_i^* can be dominated by a value $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}} \{ \Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \}$ which has a large loss, but whose value of $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ comes nowhere near the optimal value of F .

3. Support Vector Algorithm for Structured Output Spaces

So far we have not discussed how to solve the optimization problems associated with the various formulations SVM_0 , SVM_1 , SVM_2 , $SVM_1^{\Delta s}$, $SVM_1^{\Delta m}$, $SVM_2^{\Delta s}$, and $SVM_2^{\Delta m}$. The key challenge is that the size of each of these problems can be immense, since we have to deal with $n|\mathcal{Y}| - n$ margin inequalities. In many cases, $|\mathcal{Y}|$ may be extremely large, in particular, if \mathcal{Y} is a product space of some sort (e.g. in grammar learning, label sequence learning, etc.), its cardinality may grow exponentially in the description length of \mathbf{y} . This makes standard quadratic programming solvers unsuitable for this type of problem.

In the following, we will propose an algorithm that exploits the special structure of the maximum-margin problem, so that only a much smaller subset of constraints needs to be explicitly examined. We will show that the algorithm can compute arbitrary close approximations to all SVM optimization problems posed in this paper in polynomial time for a large range of structures and loss functions. Since the algorithm operates on the dual program, we will first derive the Wolfe dual for the various soft margin formulations.

3.1 Dual Programs

We will denote by $\alpha_{(i\mathbf{y})}$ the Lagrange multiplier enforcing the margin constraint for label $\mathbf{y} \neq \mathbf{y}_i$ and example $(\mathbf{x}_i, \mathbf{y}_i)$. Using standard Lagrangian duality techniques, one arrives at the following dual quadratic program (QP).

Proposition 4 *The objective of the dual problem of SVM_0 from Equation (6) is given by*

$$\Theta(\alpha) \equiv -\frac{1}{2} \sum_{i, \mathbf{y} \neq \mathbf{y}_i} \sum_{j, \bar{\mathbf{y}} \neq \mathbf{y}_j} \alpha_{(i\mathbf{y})} \alpha_{(j\bar{\mathbf{y}})} J_{(i\mathbf{y})(j\bar{\mathbf{y}})} + \sum_{i, \mathbf{y} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})},$$

where $J_{(i\mathbf{y})(j\bar{\mathbf{y}})} = \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle$. The dual QP can be formulated as

$$\alpha^* = \operatorname{argmax}_{\alpha} \Theta(\alpha), \quad \text{s.t. } \alpha \geq 0.$$

Proof (sketch) Forming the Lagrangian function and eliminating the primal variables \mathbf{w} by using the optimality condition

$$\mathbf{w}^*(\alpha) = \sum_i \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})} \delta \Psi_i(\mathbf{y})$$

directly leads to the above dual program. ■

Notice that the J function that generates the quadratic form in the dual objective can be computed from inner products involving values of Ψ , which is a simple consequence of the linearity of the inner product. J can hence be alternatively computed from a joint kernel function over $\mathcal{X} \times \mathcal{Y}$.

In the non-separable case, linear penalties introduce additional constraints, whereas the squared penalties modify the kernel function.

Proposition 5 The dual problem to SVM_1 is given by the program in Proposition 4 with additional constraints

$$\sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})} \leq \frac{C}{n}, \quad \forall i = 1, \dots, n.$$

In the following, we denote with $\delta(a, b)$ the function that returns 1 if $a = b$, and 0 otherwise.

Proposition 6 The dual problem to SVM_2 is given by the program in Proposition 4 with modified kernel function

$$J_{(i\mathbf{y})(j\bar{\mathbf{y}})} \equiv \langle \delta \Psi_i(\mathbf{y}), \delta \Psi_j(\bar{\mathbf{y}}) \rangle + \delta(i, j) \frac{n}{C}.$$

In the non-separable case with slack re-scaling, the loss function is introduced in the constraints for linear penalties and in the kernel function for quadratic penalties.

Proposition 7 The dual problem to $SVM_1^{\Delta s}$ is given by the program in Proposition 4 with additional constraints

$$\sum_{\mathbf{y} \neq \mathbf{y}_i} \frac{\alpha_{(i\mathbf{y})}}{\Delta(\mathbf{y}_i, \mathbf{y})} \leq \frac{C}{n}, \quad \forall i = 1, \dots, n.$$

Proposition 8 The dual problem to $SVM_2^{\Delta s}$ is given by the program in Proposition 4 with modified kernel function

$$J_{(i\mathbf{y})(j\bar{\mathbf{y}})} = \langle \delta \Psi_i(\mathbf{y}), \delta \Psi_j(\bar{\mathbf{y}}) \rangle + \delta(i, j) \frac{n}{C \sqrt{\Delta(\mathbf{y}_i, \mathbf{y})} \sqrt{\Delta(\mathbf{y}_j, \bar{\mathbf{y}})}}.$$

In the non-separable case with margin re-scaling, the loss function is introduced in the linear part of the objective function

Proposition 9 The dual problems to $SVM_1^{\Delta m}$ and $SVM_2^{\Delta m}$ are given by the dual problems to SVM_1 and SVM_2 with the linear part of the objective replaced by

$$\sum_{i, \mathbf{y} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})} \Delta(\mathbf{y}_i, \mathbf{y}) \quad \text{and} \quad \sum_{i, \mathbf{y} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})} \sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}$$

respectively.

3.2 Algorithm

The algorithm we propose aims at finding a small set of constraints from the full-sized optimization problem that ensures a sufficiently accurate solution. More precisely, we will construct a nested sequence of successively tighter relaxations of the original problem using a cutting plane method (Kelley, 1960), implemented as a variable selection approach in the dual formulation. Similar to its use with the Ellipsoid method (Grötschel et al., 1981; Karmarkar, 1984), we merely require a separation oracle that delivers a constraint that is violated by the current solution. We will later show that this is a valid strategy, since there always exists a polynomially sized subset of constraints so that the solution of the relaxed problem defined by this subset fulfills all constraints from the full optimization problem up to a precision of ϵ . This means, the remaining—potentially exponentially many—constraints are guaranteed to be violated by no more than ϵ , without the need for explicitly adding these constraints to the optimization problem.

We will base the optimization on the dual program formulation which has two important advantages over the primal QP. First, it only depends on inner products in the joint feature space defined by Ψ , hence allowing the use of kernel functions. Second, the constraint matrix of the dual program supports a natural problem decomposition. More specifically, notice that the constraint matrix derived for the SVM_0 and the SVM_2^* variants is diagonal, since the non-negativity constraints involve only a single α -variable at a time, whereas in the SVM_1^* case, dual variables are coupled, but the couplings only occur within a block of variables associated with the same training instance. Hence, the constraint matrix is (at least) block diagonal in all cases, where each block corresponds to a specific training instance.

Pseudo-code of the algorithm is depicted in Algorithm 1. The algorithm maintains working sets S_i for each training instance to keep track of the selected constraints which define the current relaxation. Iterating through the training examples $(\mathbf{x}_i, \mathbf{y}_i)$, the algorithm proceeds by finding the (potentially) “most violated” constraint for \mathbf{x}_i , involving some output value $\hat{\mathbf{y}}$. If the (appropriately scaled) margin violation of this constraint exceeds the current value of ξ_i by more than ϵ , the dual variable corresponding to $\hat{\mathbf{y}}$ is added to the working set. This variable selection process in the dual program corresponds to a successive strengthening of the primal problem by a cutting plane that cuts off the current primal solution from the feasible set. The chosen cutting plane corresponds to the constraint that determines the lowest feasible value for ξ_i . Once a constraint has been added, the solution is re-computed with respect to S . Alternatively, we have also devised a scheme where the optimization is restricted to S_i only, and where optimization over the full S is performed much less frequently. This can be beneficial due to the block diagonal structure of the constraint matrix, which implies that variables $\alpha_{(j\mathbf{y})}$ with $j \neq i$, $\mathbf{y} \in S_j$ can simply be “frozen” at their current values. Notice that all variables not included in their respective working set are implicitly treated as 0. The algorithm stops, if no constraint is violated by more than ϵ . With respect to the optimization in step 10, we would like to point out that in some applications the constraint selection in step 6 may be more expensive than solving the relaxed QP. Hence it may be advantageous to solve the full relaxed QP in every iteration, instead of just optimizing over a subspace of the dual variables.

The presented algorithm is implemented in the software package SVM^{struct} , available on the web at <http://svmlight.joachims.org>. Note that the SVM optimization problems from iteration to iteration differ only by a single constraint. We therefore restart the SVM optimizer from the current solution, which greatly reduces the runtime.

Algorithm 1 Algorithm for solving SVM_0 and the loss re-scaling formulations SVM_1^* and SVM_2^* .

```

1: Input:  $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n), C, \varepsilon$ 
2:  $S_i \leftarrow \emptyset$  for all  $i = 1, \dots, n$ 
3: repeat
4:   for  $i = 1, \dots, n$  do
5:     /* prepare cost function for optimization */
     set up cost function
     
$$H(\mathbf{y}) \equiv \begin{cases} 1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (SVM_0) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \Delta(\mathbf{y}_i, \mathbf{y}) & (SVM_1^{\Delta^s}) \\ \Delta(\mathbf{y}_i, \mathbf{y}) - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (SVM_1^{\Delta^m}) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \sqrt{\Delta(\mathbf{y}_i, \mathbf{y})} & (SVM_2^{\Delta^s}) \\ \sqrt{\Delta(\mathbf{y}_i, \mathbf{y})} - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (SVM_2^{\Delta^m}) \end{cases}$$

     where  $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{(j\mathbf{y}')} \delta\Psi_j(\mathbf{y}')$ .
6:     /* find cutting plane */
     compute  $\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$ 
7:     /* determine value of current slack variable */
     compute  $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$ 
8:     if  $H(\hat{\mathbf{y}}) > \xi_i + \varepsilon$  then
9:       /* add constraint to the working set */
        $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$ 
10a:      /* Variant (a): perform full optimization */
        $\alpha_S \leftarrow$  optimize the dual of  $SVM_0, SVM_1^*$  or  $SVM_2^*$  over  $S, S = \cup_i S_i$ .
10b:      /* Variant (b): perform subspace ascent */
        $\alpha_{S_i} \leftarrow$  optimize the dual of  $SVM_0, SVM_1^*$  or  $SVM_2^*$  over  $S_i$ 
12:     end if
13:   end for
14: until no  $S_i$  has changed during iteration
    
```

A convenient property of both variants of the cutting plane algorithm is that they have a very general and well-defined interface independent of the choice of Ψ and Δ . To apply the algorithm, it is sufficient to implement the feature mapping $\Psi(\mathbf{x}, \mathbf{y})$ (either explicitly or via a joint kernel function), the loss function $\Delta(\mathbf{y}_i, \mathbf{y})$, as well as the maximization in step 6. All of those, in particular the constraint/cut selection method, are treated as black boxes. While the modeling of $\Psi(\mathbf{x}, \mathbf{y})$ and $\Delta(\mathbf{y}_i, \mathbf{y})$ is typically straightforward, solving the maximization problem for constraint selection typically requires exploiting the structure of Ψ for output spaces that can not be dealt with by exhaustive search.

In the slack re-scaling setting, it turns out that for a given example $(\mathbf{x}_i, \mathbf{y}_i)$ we need to identify the maximum over

$$\hat{\mathbf{y}} \equiv \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \{(1 - \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle) \Delta(\mathbf{y}_i, \mathbf{y})\}.$$

We will discuss several cases for how to solve this problem in Section 4. Typically, it can be solved by an appropriate modification of the prediction problem in Equation (1), which recovers f from F . For example, in the case of grammar learning with the F_1 score as the loss function via $\Delta(\mathbf{y}_i, \mathbf{y}) = (1 - F_1(\mathbf{y}_i, \mathbf{y}))$, the maximum can be computed using a modified version of the CKY algorithm. More generally, in cases where $\Delta(\mathbf{y}_i, \cdot)$ only takes on a finite number of values, a generic strategy is a two stage approach, where one first computes the maximum over those \mathbf{y} for which the loss is constant, $\Delta(\mathbf{y}_i, \mathbf{y}) = \text{const}$, and then maximizes over the finite number of levels.

In the margin re-scaling setting, one needs to solve the maximization problem

$$\hat{\mathbf{y}} \equiv \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \{\Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle\}. \quad (9)$$

In cases where the loss function has an additive decomposition that is compatible with the feature map, one can fold the loss function contribution into the weight vector $\langle \mathbf{w}', \delta \Psi_i(\mathbf{y}) \rangle = \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle - \Delta(\mathbf{y}_i, \mathbf{y})$ for some \mathbf{w}' . This means the class of cost functions defined by $F(\mathbf{x}, \cdot; \mathbf{w})$ and $F(\mathbf{x}, \cdot; \mathbf{w}) - \Delta(\mathbf{y}, \cdot)$ may actually be identical.

The algorithm for the zero-one loss is a special case of either algorithm. We need to identify the highest scoring \mathbf{y} that is incorrect,

$$\hat{\mathbf{y}} \equiv \operatorname{argmax}_{\mathbf{y} \neq \mathbf{y}_i} \{1 - \langle \mathbf{w}, \delta \Psi_i(\mathbf{y}) \rangle\}.$$

It is therefore sufficient to identify the best solution $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$ as well as the second best solution $\tilde{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y} \setminus \{\hat{\mathbf{y}}\}} \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$. The second best solution is necessary to detect margin violations in cases where $\hat{\mathbf{y}} = \mathbf{y}_i$, but $\langle \mathbf{w}, \delta \Psi_i(\tilde{\mathbf{y}}) \rangle < 1$. This means that for all problems where we can solve the inference problem in Equation (1) for the top two \mathbf{y} , we can also apply our learning algorithms with the zero-one loss. In the case of grammar learning, for example, we can use any existing parser that returns the two highest scoring parse trees.

We will now proceed by analyzing the presented family of algorithms. In particular, we will show correctness and sparse approximation properties, as well as bounds on the runtime complexity.

3.3 Correctness and Complexity of the Algorithm

What we would like to accomplish first is to obtain a lower bound on the achievable improvement of the dual objective by selecting a single variable $\alpha_{(i\tilde{\mathbf{y}})}$ and adding it to the dual problem (cf. step 10 in Algorithm 1). While this is relatively straightforward when using quadratic penalties, the SVM₁ formulation introduces an additional complication in the form of upper bounds on non-overlapping subsets of variables, namely the set of variables $\alpha_{(i\tilde{\mathbf{y}})}$ in the current working set that correspond to the same training instance. Hence, we may not be able to answer the above question by optimizing over $\alpha_{(i\tilde{\mathbf{y}})}$ alone, but rather have to deal with a larger optimization problem over a whole subspace. In order to derive useful bounds, it suffices to restrict attention to simple one-dimensional families of solutions that are defined by improving an existing solution along a specific direction η . Proving

that one can make sufficient progress along a specific direction, clearly implies that one can make at least that much progress by optimizing over a larger subspace that includes the direction η . A first step towards executing this idea is the following lemma.

Lemma 10 *Let J be a symmetric, positive semi-definite matrix, and define a concave objective in α*

$$\Theta(\alpha) = -\frac{1}{2}\alpha'J\alpha + \langle \mathbf{h}, \alpha \rangle,$$

which we assume to be bounded from above. Assume that a solution α^o and an optimization direction η is given such that $\langle \nabla\Theta(\alpha^o), \eta \rangle > 0$. Then optimizing Θ starting from α^o along the chosen direction η will increase the objective by

$$\max_{\beta > 0} \{\Theta(\alpha^o + \beta\eta)\} - \Theta(\alpha^o) = \frac{1}{2} \frac{\langle \nabla\Theta(\alpha^o), \eta \rangle^2}{\eta'J\eta} > 0.$$

Proof *The difference obtained by a particular β is given by*

$$\delta\Theta(\beta) \equiv \beta \left[\langle \nabla\Theta(\alpha^o), \eta \rangle - \frac{\beta}{2}\eta'J\eta \right],$$

as can be verified by elementary algebra. Solving for β one arrives at

$$\frac{d}{d\beta}\delta\Theta = 0 \iff \beta^* = \frac{\langle \nabla\Theta(\alpha^o), \eta \rangle}{\eta'J\eta}.$$

Notice that this requires $\eta'J\eta > 0$. Obviously, the positive semi-definiteness of J guarantees $\eta'J\eta \geq 0$ for any η . Moreover $\eta'J\eta = 0$ together with $\langle \nabla\Theta(\alpha^o), \eta \rangle > 0$ would imply that $\lim_{\beta \rightarrow \infty} \Theta(\alpha^o + \beta\eta) = \infty$, which is in contradiction with the assumption that Θ is bounded. Plugging the value for β^ back into the above expression for $\delta\Theta$ yields the claim. ■*

Corollary 11 *Under the same assumption as in Lemma 10 and for the special case of an optimization direction $\eta = e_r$, the objective improves by*

$$\delta\Theta(\beta^*) = \frac{1}{2J_{rr}} \left(\frac{\partial\Theta}{\partial\alpha_r} \right)^2 > 0.$$

Proof *Notice that $\eta = e_r$ implies $\langle \nabla\Theta, \eta \rangle = \frac{\partial\Theta}{\partial\alpha_r}$ and $\eta'J\eta = J_{rr}$. ■*

Corollary 12 *Under the same assumptions as in Lemma 10 and enforcing the constraint $\beta \leq D$ for some $D > 0$, the objective improves by*

$$\max_{0 < \beta \leq D} \{\Theta(\alpha^o + \beta\eta)\} - \Theta(\alpha^o) = \begin{cases} \frac{\langle \nabla\Theta(\alpha^o), \eta \rangle^2}{2\eta'J\eta} & \text{if } \langle \nabla\Theta(\alpha^o), \eta \rangle \leq D\eta'J\eta \\ D\langle \nabla\Theta(\alpha^o), \eta \rangle - \frac{D^2}{2}\eta'J\eta & \text{otherwise} \end{cases}.$$

Moreover, the improvement can be upper bounded by

$$\max_{0 < \beta \leq D} \{\Theta(\alpha^o + \beta\eta)\} - \Theta(\alpha^o) \geq \frac{1}{2} \min \left\{ D, \frac{\langle \nabla\Theta(\alpha^o), \eta \rangle}{\eta'J\eta} \right\} \langle \nabla\Theta(\alpha^o), \eta \rangle.$$

Proof We distinguish two cases of either $\beta^* \leq D$ or $\beta^* > D$. In the first case, we can simply apply lemma 10 since the additional constraint is inactive and does not change the solution. In the second case, the concavity of Θ implies that $\beta = D$ achieves the maximum of $\delta\Theta$ over the constrained range. Plugging in this result for β^* into $\delta\Theta$ yields the second case in the claim.

Finally, the bound is obtained by exploiting that in the second case

$$\beta^* > D \iff D < \frac{\langle \nabla\Theta(\alpha^o), \eta \rangle}{\eta'J\eta}.$$

Replacing one of the D factors in the D^2 term of the second case with this bound yields an upper bound. The first (exact) case and the bound in the second case can be compactly combined as shown in the formula of the claim. ■

Corollary 13 Under the same assumption as in Corollary 12 and for the special case of a single-coordinate optimization direction $\eta = e_r$, the objective improves at least by

$$\max_{0 < \beta \leq D} \Theta(\alpha^o + \beta e_r) - \Theta(\alpha^o) \geq \frac{1}{2} \min \left\{ D, \frac{\frac{\partial\Theta}{\partial\alpha_r}(\alpha^o)}{J_{rr}} \right\} \frac{\partial\Theta}{\partial\alpha_r}(\alpha^o)$$

Proof Notice that $\eta = e_r$ implies $\langle \nabla\Theta, \eta \rangle = \frac{\partial\Theta}{\partial\alpha_r}$ and $\eta'J\eta = J_{rr}$. ■

We now apply the above lemma and corollaries to the four different SVM formulations, starting with the somewhat simpler squared penalty case.

Proposition 14 (SVM₂^{Δs}) For SVM₂^{Δs} step 10 in Algorithm 1 the improvement $\delta\Theta$ of the dual objective is lower bounded by

$$\delta\Theta \geq \frac{1}{2} \frac{\varepsilon^2}{\Delta_i R_i^2 + \frac{n}{C}}, \quad \text{where } \Delta_i \equiv \max_{\mathbf{y}} \{\Delta(\mathbf{y}_i, \mathbf{y})\} \text{ and } R_i \equiv \max_{\mathbf{y}} \{\|\delta\Psi_i(\mathbf{y})\|\}.$$

Proof Using the notation in Algorithm 1 one can apply Corollary 11 with multi-index $r = (i\hat{\mathbf{y}})$, $h = 1$, and J such that

$$J_{(i\hat{\mathbf{y}})(j\hat{\mathbf{y}})} = \langle \delta\Psi_i(\hat{\mathbf{y}}), \delta\Psi_j(\mathbf{y}) \rangle + \frac{\delta(i, j)n}{C\sqrt{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})}\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}}.$$

Notice that the partial derivative of Θ with respect to $\alpha_{(i\hat{\mathbf{y}})}$ is given by

$$\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\alpha^o) = 1 - \sum_{j:\mathbf{y}} \alpha_{(j\hat{\mathbf{y}}}^o J_{(i\hat{\mathbf{y}})(j\hat{\mathbf{y}})} = 1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}}) \rangle - \frac{\xi_i^*}{\sqrt{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})}},$$

since the optimality equations for the primal variables yield the identities

$$\mathbf{w}^* = \sum_{j,\mathbf{y}} \alpha_{(j\mathbf{y})}^o \delta\Psi_j(\mathbf{y}), \quad \text{and} \quad \xi_i^* = \sum_{\mathbf{y} \neq \mathbf{y}_i} \frac{n\alpha_{(i\mathbf{y})}^o}{C\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}}.$$

Now, applying the condition of step 10, namely $\sqrt{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})}(1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}}) \rangle) > \xi_i^* + \varepsilon$, leads to the bound

$$\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\alpha^o) \geq \frac{\varepsilon}{\sqrt{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})}}.$$

Finally, $J_{rr} = \|\delta\Psi_i(\hat{\mathbf{y}})\|^2 + \frac{n}{C\Delta(\mathbf{y}_i, \hat{\mathbf{y}})}$ and inserting this expression and the previous bound into the expression from Corollary 11 yields

$$\frac{1}{2J_{rr}} \left(\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}} \right)^2 \geq \frac{\varepsilon^2}{2(\Delta(\mathbf{y}_i, \hat{\mathbf{y}})\|\delta\Psi_i(\hat{\mathbf{y}})\|^2 + \frac{n}{C})} \geq \frac{\varepsilon^2}{2(\Delta_i R_i^2 + \frac{n}{C})}.$$

The claim follows by observing that jointly optimizing over a set of variables that include α_r can only further increase the value of the dual objective. \blacksquare

Proposition 15 (SVM₂^{△*m*}) For SVM₂^{△*m*} step 10 in Algorithm 1 the improvement $\delta\Theta$ of the dual objective is lower bounded by

$$\delta\Theta \geq \frac{1}{2} \frac{\varepsilon^2}{R_i^2 + \frac{n}{C}}, \quad \text{where} \quad R_i = \max_{\mathbf{y}} \|\delta\Psi_i(\mathbf{y})\|.$$

Proof By re-defining $\delta\tilde{\Psi}_i(\mathbf{y}) \equiv \frac{\delta\Psi_i(\mathbf{y})}{\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}}$ we are back to Proposition 14 with

$$\max_{\mathbf{y}} \{\Delta(\mathbf{y}_i, \mathbf{y})\|\delta\tilde{\Psi}_i(\mathbf{y})\|^2\} = \max_{\mathbf{y}} \{\|\delta\Psi_i(\mathbf{y})\|^2\} = R_i^2,$$

since

$$\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq \sqrt{\Delta(\mathbf{y}_i, \mathbf{y})} - \xi_i \iff \langle \mathbf{w}, \delta\tilde{\Psi}_i(\mathbf{y}) \rangle \geq 1 - \frac{\xi_i}{\sqrt{\Delta(\mathbf{y}_i, \mathbf{y})}}.$$

\blacksquare

Proposition 16 (SVM₁^{△*s*}) For SVM₁^{△*s*} step 10 in Algorithm 1 the improvement $\delta\Theta$ of the dual objective is lower bounded by

$$\delta\Theta \geq \min \left\{ \frac{C\varepsilon}{2n}, \frac{\varepsilon^2}{8\Delta_i^2 R_i^2} \right\} \quad \text{where} \quad \Delta_i = \max_{\mathbf{y}} \{\Delta(\mathbf{y}_i, \mathbf{y})\} \quad \text{and} \quad R_i = \max_{\mathbf{y}} \{\|\delta\Psi_i(\mathbf{y})\|\}.$$

Proof

Case I:

If the working set does not contain an element $(i\mathbf{y})$, then we can optimize over $\alpha_{(i\hat{\mathbf{y}})}$ under the constraint that $\alpha_{(i\hat{\mathbf{y}})} \leq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}) \frac{C}{n} = D$. Notice that

$$\frac{\partial \Theta}{\partial \alpha_{(i\hat{\mathbf{y}})}}(\alpha^o) = 1 - \langle \mathbf{w}^*, \delta \Psi_i(\hat{\mathbf{y}}) \rangle > \frac{\xi_i^* + \varepsilon}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \geq \frac{\varepsilon}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})},$$

where the first inequality follows from the pre-condition for selecting $(i\hat{\mathbf{y}})$ and the last one from $\xi_i^* \geq 0$. Moreover, notice that $J_{(i\hat{\mathbf{y}})(i\hat{\mathbf{y}})} \leq R_i^2$. Evoking Corollary 13 with the obvious identifications yields

$$\begin{aligned} \delta \Theta &\geq \frac{1}{2} \min \left\{ D, \frac{1}{J_{rr}} \frac{\partial \Theta}{\partial \alpha_{(i\hat{\mathbf{y}})}}(\alpha^o) \right\} \frac{\partial \Theta}{\partial \alpha_{(i\hat{\mathbf{y}})}}(\alpha^o) \\ &> \frac{1}{2} \min \left\{ \frac{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})C}{n}, \frac{\varepsilon}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})R_i^2} \right\} \frac{\varepsilon}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} = \min \left\{ \frac{C\varepsilon}{2n}, \frac{\varepsilon^2}{2R_i^2 \Delta(\mathbf{y}_i, \hat{\mathbf{y}})^2} \right\} \end{aligned}$$

The second term can be further bounded to yield the claim.

Case II:

If there are already active constraints for instance \mathbf{x}_i in the current working set, i.e. $S_i \neq \emptyset$, then we may need to reduce dual variables $\alpha_{(i\mathbf{y})}$ in order to get some slack for increasing the newly added $\alpha_{(i\hat{\mathbf{y}})}$. We thus investigate search directions η such that $\eta_{(i\hat{\mathbf{y}})} = 1$, $\eta_{(i\mathbf{y})} = -\frac{\alpha_{(i\mathbf{y})}}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \frac{n}{C} \leq 0$ for $\mathbf{y} \in S_i$, and $\eta_{(j\mathbf{y}')} = 0$ in all other cases. For such η , we guarantee that $\alpha^o + \beta \eta \geq 0$ since $\beta \leq \frac{C}{n} \Delta(\mathbf{y}_i, \hat{\mathbf{y}})$. In finding a suitable direction to derive a good bound, we have two (possibly conflicting) goals. First of all, we want the directional derivative to be positively bounded away from zero. Notice that

$$\langle \nabla \Theta(\alpha^o), \eta \rangle = \sum_{\mathbf{y}} \eta_{(i\mathbf{y})} (1 - \langle \mathbf{w}^*, \delta \Psi_i(\mathbf{y}) \rangle).$$

Furthermore, by the restrictions imposed on η , $\eta_{(i\mathbf{y})} < 0$ implies that the respective constraint is active and hence $\Delta(\mathbf{y}_i, \mathbf{y}) (1 - \langle \mathbf{w}^*, \delta \Psi_i(\mathbf{y}) \rangle) = \xi_i^*$. Moreover the pre-condition of step 10 ensures that $\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) (1 - \langle \mathbf{w}^*, \delta \Psi_i(\hat{\mathbf{y}}) \rangle) = \xi_i^* + \delta$ where $\delta \geq \varepsilon > 0$. Hence

$$\langle \nabla \Theta(\alpha^o), \eta \rangle = \frac{\xi_i^*}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \left(1 - \frac{n}{C} \sum_{\mathbf{y}} \frac{\alpha_{(i\mathbf{y})}^o}{\Delta(\mathbf{y}_i, \mathbf{y})} \right) + \frac{\delta}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \geq \frac{\varepsilon}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})}.$$

The second goal is to make sure the curvature along the chosen direction is not too large.

$$\begin{aligned} \eta' J \eta &= J_{(i\hat{\mathbf{y}})(i\hat{\mathbf{y}})} - 2 \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \frac{\alpha_{(i\mathbf{y})}^o}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \frac{n}{C} J_{(i\hat{\mathbf{y}})(i\mathbf{y})} + \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \sum_{\mathbf{y}' \neq \hat{\mathbf{y}}} \frac{\alpha_{(i\mathbf{y})}^o}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \frac{n}{C} \frac{\alpha_{(i\mathbf{y}')}^o}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \frac{n}{C} J_{(i\mathbf{y})(i\mathbf{y}')} \\ &\leq R_i^2 + 2 \frac{nR_i^2}{C\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \alpha_{(i\mathbf{y})}^o + \frac{n^2 R_i^2}{C^2 \Delta(\mathbf{y}_i, \hat{\mathbf{y}})^2} \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \sum_{\mathbf{y}' \neq \hat{\mathbf{y}}} \alpha_{(i\mathbf{y})}^o \alpha_{(i\mathbf{y}')}^o \\ &\leq R_i^2 + 2 \frac{R_i^2 \Delta_i}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} + \frac{R_i^2 \Delta_i^2}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})^2} \leq \frac{4R_i^2 \Delta_i^2}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})^2}. \end{aligned}$$

This follows from the fact that $\sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \alpha_{(i\mathbf{y})}^o \leq \Delta_i \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \frac{\alpha_{(i\mathbf{y})}^o}{\Delta(\mathbf{y}_i, \mathbf{y})} \leq \frac{C\Delta_i}{n}$. Evoking Corollary 12 yields

$$\begin{aligned} \delta\Theta &\geq \frac{1}{2} \min \left\{ D, \frac{\langle \nabla\Theta(\alpha^o), \eta \rangle}{\eta' J \eta} \right\} \langle \nabla\Theta(\alpha^o), \eta \rangle \\ &\geq \frac{1}{2} \min \left\{ \frac{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})C}{n}, \frac{\frac{\varepsilon}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})}}{4R_i^2 \Delta_i^2} \right\} \frac{\varepsilon}{\Delta(\mathbf{y}_i, \hat{\mathbf{y}})} = \min \left\{ \frac{C\varepsilon}{2n}, \frac{\varepsilon^2}{8R_i^2 \Delta_i^2} \right\} \end{aligned}$$

■

Proposition 17 ($\text{SVM}_1^{\Delta m}$) For $\text{SVM}_1^{\Delta m}$ step 10 in Algorithm 1 the improvement $\delta\Theta$ of the dual objective is lower bounded by

$$\delta\Theta \geq \frac{\varepsilon^2}{8R_i^2}, \quad \text{where } R_i = \max_{\mathbf{y}} \|\delta\Psi_i(\mathbf{y})\|.$$

Proof By re-defining $\delta\tilde{\Psi}_i(\mathbf{y}) \equiv \frac{\delta\Psi_i(\mathbf{y})}{\Delta(\mathbf{y}_i, \mathbf{y})}$ we are back to Proposition 16 with

$$\max_{\mathbf{y}} \{ \Delta(\mathbf{y}_i, \mathbf{y})^2 \|\delta\tilde{\Psi}_i(\mathbf{y})\|^2 \} = \max_{\mathbf{y}} \{ \|\delta\Psi_i(\mathbf{y})\|^2 \} = R_i^2,$$

since

$$\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \iff \langle \mathbf{w}, \delta\tilde{\Psi}_i(\mathbf{y}) \rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{y}_i, \mathbf{y})}.$$

■

This leads to the following polynomial bound on the maximum size of S .

Theorem 18 With $\bar{R} = \max_i R_i$, $\bar{\Delta} = \max_i \Delta_i$ and for a given $\varepsilon > 0$, Algorithm 1 terminates after incrementally adding at most

$$\max \left\{ \frac{2n\bar{\Delta}}{\varepsilon}, \frac{8C\bar{\Delta}^3\bar{R}^2}{\varepsilon^2} \right\}, \max \left\{ \frac{2n\bar{\Delta}}{\varepsilon}, \frac{8C\bar{\Delta}\bar{R}^2}{\varepsilon^2} \right\}, \frac{C\bar{\Delta}^2\bar{R}^2 + n\bar{\Delta}}{\varepsilon^2} \text{ and } \frac{C\bar{\Delta}\bar{R}^2 + n\bar{\Delta}}{\varepsilon^2}$$

constraints to the working set S for the $\text{SVM}_1^{\Delta s}$, $\text{SVM}_1^{\Delta m}$, $\text{SVM}_2^{\Delta s}$ and $\text{SVM}_2^{\Delta m}$ respectively.

Proof With $S = \emptyset$ the optimal value of the dual is 0. In each iteration a constraint is added that is violated by at least ε , provided such a constraint exists. After solving the S -relaxed QP in step 10, the objective will increase by at least the amounts suggested by Propositions 16, 17, 14 and 15 respectively. Hence after t constraints, the dual objective will be at least t times these increments. The result follows from the fact that the dual objective is upper bounded by the minimum of the primal, which in turn can be bounded by $C\bar{\Delta}$ and $\frac{1}{2}C\bar{\Delta}$ for SVM_1^* and SVM_2^* respectively. ■

Note that the number of constraints in S does not depend on $|\mathcal{Y}|$. This is crucial, since $|\mathcal{Y}|$ is exponential or infinite for many interesting problems. For problems where step 6 can be computed in polynomial time, the overall algorithm has a runtime polynomial in $n, \bar{R}, \bar{\Delta}, 1/\varepsilon$, since at least one constraint will be added while cycling through all n instances and since step 10 is polynomial. This shows that the algorithm considers only a small number of constraints, if one allows an extra ε slack, and that the solution is correct up to an approximation that depends on the precision parameter ε . The upper bound on the number of active constraints in such an approximate solution depends on the chosen representation, more specifically, we need to upper bound the difference vectors $\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \bar{\mathbf{y}})\|^2$ for arbitrary $\mathbf{y}, \bar{\mathbf{y}} \in \mathcal{Y}$. In the following, we will thus make sure that suitable upper bounds are available.

4. Specific Problems and Special Cases

In the sequel, we will discuss a number of interesting special cases of the general scenario outlined in the previous section. To model each particular problem and to be able to run the algorithm and bound its complexity, we need to examine the following three questions for each case:

- *Modeling*: How can we define suitable feature maps $\Psi(\mathbf{x}, \mathbf{y})$ for specific problems?
- *Algorithms*: How can we compute the required maximization over \mathcal{Y} for given \mathbf{x} ?
- *Sparseness*: How can we bound $\|\Psi(\mathbf{x}, \mathbf{y}) - \Psi(\mathbf{x}, \mathbf{y}')\|$?

4.1 Multiclass Classification

A special case of Equation (1) is winner-takes-all (WTA) multiclass classification, where $\mathcal{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_K\}$ and $\mathbf{w} = (\mathbf{v}'_1, \dots, \mathbf{v}'_K)'$ is a stack of vectors, \mathbf{v}_k being a weight vector associated with the k -th class \mathbf{y}_k . The WTA rule is given by

$$f(\mathbf{x}) = \arg \max_{\mathbf{y}_k \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}), \quad F(\mathbf{x}, \mathbf{y}_k; \mathbf{w}) = \langle \mathbf{v}_k, \Phi(\mathbf{x}) \rangle. \quad (10)$$

Here $\Phi(\mathbf{x}) \in \mathbb{R}^D$ denotes an arbitrary feature representation of the inputs, which in many cases may be defined implicitly via a kernel function.

4.1.1 MODELING

The above decision rule can be equivalently represented by making use of a joint feature map as follows. First of all, we define the canonical (binary) representation of labels $\mathbf{y} \in \mathcal{Y}$ by unit vectors

$$\Lambda^c(\mathbf{y}) \equiv (\delta(\mathbf{y}_1, \mathbf{y}), \delta(\mathbf{y}_2, \mathbf{y}), \dots, \delta(\mathbf{y}_K, \mathbf{y}))' \in \{0, 1\}^K, \quad (11)$$

so that $\langle \Lambda^c(\mathbf{y}), \Lambda^c(\mathbf{y}') \rangle = \delta(\mathbf{y}, \mathbf{y}')$. It will turn out to be convenient to use direct tensor products \otimes to combine feature maps over \mathcal{X} and \mathcal{Y} . In general, we thus define the \otimes -operation in the following manner

$$\otimes : \mathbb{R}^D \times \mathbb{R}^K \rightarrow \mathbb{R}^{D \cdot K}, \quad (\mathbf{a} \otimes \mathbf{b})_{i+(j-1)D} \equiv a_i \cdot b_j.$$

Now we can define a joint feature map for the multiclass problem by

$$\Psi(\mathbf{x}, \mathbf{y}) \equiv \Phi(\mathbf{x}) \otimes \Lambda^c(\mathbf{y}). \quad (12)$$

It is easy to show that this results in an equivalent formulation of the multiclass WTA as expressed in the following proposition.

Proposition 19 $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$, where F is defined in Equation (10) and Ψ in Equation (12).

Proof For all $\mathbf{y}_k \in \mathcal{Y}$: $\langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}_k) \rangle = \sum_{r=1}^{D-K} w_r \psi_r(\mathbf{x}, \mathbf{y}_k) = \sum_{j=1}^K \sum_{d=1}^D v_{jd} \phi_d(\mathbf{x}) \delta(j, k) = \sum_{d=1}^D v_{kd} \phi_d(\mathbf{x}) = \langle \mathbf{v}_k, \Phi(\mathbf{x}) \rangle$. ■

4.1.2 ALGORITHMS

It is usually assumed that the number of classes K in simple multiclass problems is small enough, so that an exhaustive search can be performed to maximize any objective over \mathcal{Y} . Similarly, we can find the second best $\mathbf{y} \in \mathcal{Y}$.

4.1.3 SPARSENESS

In order to bound the norm of the difference feature vectors, we prove the following simple result.

Proposition 20 Define $R_i \equiv \|\Phi(\mathbf{x}_i)\|$. Then $\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}')\|^2 \leq 2R_i^2$.

Proof

$$\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}')\|^2 \leq \|\Psi(\mathbf{x}_i, \mathbf{y})\|^2 + \|\Psi(\mathbf{x}_i, \mathbf{y}')\|^2 = 2\|\Phi(\mathbf{x}_i)\|^2,$$

where the first step follows from the Cauchy-Schwarz inequality and the second step exploits the sparseness of Λ^c . ■

4.2 Multiclass Classification with Output Features

The first generalization we propose is to make use of more interesting output features Λ than the canonical representation in Equation (11). Apparently, we could use the same approach as in Equation (12) to define a joint feature function, but use a more general form for Λ .

4.2.1 MODELING

We first show that for any joint feature map Ψ constructed via the direct tensor product \otimes the following relation holds:

Proposition 21 For $\Psi = \Phi \otimes \Lambda$ the inner product can be written as

$$\langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', \mathbf{y}') \rangle = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \cdot \langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle.$$

Proof *By simple algebra*

$$\begin{aligned} \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', \mathbf{y}') \rangle &= \sum_{r=1}^{D \cdot K} \sum_{s=1}^{D \cdot K} \psi_r(\mathbf{x}, \mathbf{y}) \psi_s(\mathbf{x}', \mathbf{y}') = \sum_{d=1}^D \sum_{k=1}^K \sum_{d'=1}^D \sum_{k'=1}^K \phi_d(\mathbf{x}) \lambda_k(\mathbf{y}) \phi_{d'}(\mathbf{x}') \lambda_{k'}(\mathbf{y}') \\ &= \sum_{d=1}^D \sum_{d'=1}^D \phi_d(\mathbf{x}) \phi_{d'}(\mathbf{x}') \sum_{k=1}^K \sum_{k'=1}^K \lambda_k(\mathbf{y}) \lambda_{k'}(\mathbf{y}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \cdot \langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle. \end{aligned}$$

■

This implies that for feature maps Φ that are implicitly defined via kernel functions K , $K(\mathbf{x}, \mathbf{x}') \equiv \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$, one can define a joint kernel function as follows:

$$J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', \mathbf{y}') \rangle = \langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle K(\mathbf{x}, \mathbf{x}').$$

Of course, nothing prevents us from expressing the inner product in output space via yet another kernel function $L(\mathbf{y}, \mathbf{y}') = \langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle$. Notice that the kernel L is simply the identity in the standard multiclass case. How can this kernel be chosen in concrete cases? It basically may encode any type of prior knowledge one might have about the similarity between classes. It is illuminating to note the following proposition.

Proposition 22 *Define $\Psi(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \otimes \Lambda(\mathbf{y})$ with $\Lambda(\mathbf{y}) \in \mathbb{R}^R$; then the discriminant function $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ can be written as*

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{r=1}^R \lambda_r(\mathbf{y}) \langle \mathbf{v}_r, \Phi(\mathbf{x}) \rangle,$$

where $\mathbf{w} = (\mathbf{v}'_1, \dots, \mathbf{v}'_R)'$ is the stack of vectors $\mathbf{v}_r \in \mathbb{R}^D$, one vector for each basis function of Λ .

Proof

$$\begin{aligned} \sum_{r=1}^R \lambda_r(\mathbf{y}) \sum_{d=1}^D v_{rd} \phi_d(\mathbf{x}) &= \sum_{r=1}^R \sum_{d=1}^D w_{D \cdot (d-1) + r} \lambda_r(\mathbf{y}) \phi_d(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \otimes \Lambda(\mathbf{y}) \rangle \\ &= \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle = F(\mathbf{x}, \mathbf{y}; \mathbf{w}). \end{aligned}$$

■

We can give this a simple interpretation: For each output feature λ_r a corresponding weight vector \mathbf{v}_r is introduced. The discriminant function can then be represented as a weighted sum of contributions coming from the different features. In particular, in the case of binary features $\Lambda : \mathcal{Y} \rightarrow \{0, 1\}^R$, this will simply be a sum over all contributions $\langle \mathbf{v}_r, \Phi(\mathbf{x}) \rangle$ of features that are active for the class \mathbf{y} , i.e. for which $\lambda_r(\mathbf{y}) = 1$.

It is also important to note that the orthogonal representation provides a maximally large hypothesis class and that nothing can be gained in terms of representational power by including *additional* features.

Corollary 23 Assume a mapping $\Lambda(\mathbf{y}) = (\tilde{\Lambda}(\mathbf{y})', \Lambda^c(\mathbf{y})')'$, $\tilde{\Lambda}(\mathbf{y}) \in \mathbb{R}^R$ and define $\tilde{\Psi}(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \otimes \Lambda(\mathbf{y})$ and $\Psi(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \otimes \Lambda^c(\mathbf{y})$. Now, for every $\tilde{\mathbf{w}}$ there is \mathbf{w} such that $\langle \tilde{\mathbf{w}}, \tilde{\Psi}(\mathbf{x}, \mathbf{y}) \rangle = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$ and vice versa.

Proof Applying Proposition 22 twice it follows that

$$\langle \tilde{\mathbf{w}}, \tilde{\Psi}(\mathbf{x}, \mathbf{y}) \rangle = \sum_{r=1}^{R+K} \lambda_r(\mathbf{y}) \langle \tilde{\mathbf{v}}_r, \Phi(\mathbf{x}) \rangle = \left\langle \sum_{r=1}^{R+K} \lambda_r(\mathbf{y}) \tilde{\mathbf{v}}_r, \Phi(\mathbf{x}) \right\rangle = \langle \mathbf{v}_y, \Phi(\mathbf{x}) \rangle = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle.$$

where we have defined $\mathbf{v}_y = \sum_{r=1}^{R+K} \lambda_r(\mathbf{y}) \tilde{\mathbf{v}}_r$. The reverse direction is trivial and requires setting $\tilde{\mathbf{v}}_r = 0$ for $r = 1, \dots, R$. ■

In the light of this corollary, we would like to emphasize that the rationale behind the use of class features is not to increase the representational power of the hypothesis space, but to re-parameterize (or even constrain) the hypothesis space such that a more suitable representation for \mathcal{Y} is produced. We would like to *generalize across classes* as we want to generalize across input patterns in the standard formulation of classification problems. Obviously, orthogonal representations (corresponding to diagonal kernels) will provide no generalization whatsoever across different classes \mathbf{y} . The choice of a good output feature map Λ is thus expected to provide an inductive bias, namely that learning can occur across a set of classes sharing a common property.

Let us discuss some special cases of interest.

Classification with Taxonomies Assume that class labels \mathbf{y} are arranged in a taxonomy. We will define a taxonomy as a set of elements $\mathcal{Z} \supseteq \mathcal{Y}$ equipped with a partial order \prec . The partially ordered set (\mathcal{Z}, \prec) might, for example, represent a tree or a lattice. Now we can define binary features for classes as follows: Associate one feature $\lambda_{\mathbf{z}}$ with every element in \mathcal{Z} according to

$$\lambda_{\mathbf{z}}(\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{y} \prec \mathbf{z} \text{ or } \mathbf{y} = \mathbf{z} \\ 0 & \text{otherwise.} \end{cases}$$

This includes multiclass classification as a special case of an unordered set $\mathcal{Z} = \mathcal{Y}$. In general, however, the features $\lambda_{\mathbf{z}}$ will be “shared” by all classes below \mathbf{z} , e.g. all nodes \mathbf{y} in the subtree rooted at \mathbf{z} in the case of a tree. One may also introduce a relative weight $\beta_{\mathbf{z}}$ for every feature and define a β -weighted (instead of binary) output feature map $\tilde{\Lambda}$ as $\tilde{\lambda}_{\mathbf{z}} = \beta_{\mathbf{z}} \lambda_{\mathbf{z}}$. If we reflect upon the implication of this definition in the light of Proposition 22, one observes that this effectively introduces a weight vector $\mathbf{v}_{\mathbf{z}}$ for every element of \mathcal{Z} , i.e. for every node in the hierarchy.

Learning with Textual Class Descriptions As a second motivating example, we consider problems where classes are characterized by short glosses, blurbs or other textual descriptions. We would like to exploit the fact that classes sharing some descriptors are likely to be similar, in order to specify a suitable inductive bias. This can be achieved, for example, by associating a feature λ with every keyword used to describe classes, in addition to the class identity. Hence standard vector space models like term-frequency or idf representations can be applied to model classes and the inner product $\langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle$ then defines a similarity measure between classes corresponding to the standard cosine-measure used in information retrieval.

Learning with Class Similarities The above example can obviously be generalized to any situation, where we have access to a positive definite similarity function for pairs of classes. To come up with suitable similarity functions is part of the domain model—very much like determining a good representation of the inputs—and we assume here that it is given.

4.2.2 ALGORITHMS

As in the multiclass case, we assume that the number of classes is small enough to perform an exhaustive search.

4.2.3 SPARSENESS

Proposition 20 can be generalized in the following way:

Proposition 24 Define $R_i \equiv \|\Phi(\mathbf{x}_i)\|$ and $S \equiv \max_{\mathbf{y} \in \mathcal{Y}} \|\Lambda(\mathbf{y})\|$ then $\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}')\|^2 \leq 2R_i^2 S^2$ for all $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$.

Proof $\langle \Psi(\mathbf{x}_i, \mathbf{y}), \Psi(\mathbf{x}_i, \mathbf{y}') \rangle = \|\Phi(\mathbf{x}_i)\|^2 \cdot \|\Lambda(\mathbf{y})\|^2 \leq R_i^2 S^2$. In the last step, we have used Proposition 21. \blacksquare

4.3 Label Sequence Learning

The next problem we would like to formulate in the joint feature map framework is the problem of label sequence learning, or sequence segmentation/annotation. Here, the goal is to predict a label sequence $\mathbf{y} = (y^1, \dots, y^T)$ for a given observation sequence $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^T)$. In order to simplify the presentation, let us assume all sequences are of the same length T . Let us denote by Σ the set of possible labels for each individual variable y^t , i.e. $\mathcal{Y} = \Sigma^T$. Hence each sequence of labels is considered to be a class of its own, resulting in a multiclass classification problem with $|\Sigma|^T$ different classes. To model label sequence learning in this manner would of course not be very useful, if one were to apply standard multiclass classification methods. However, this can be overcome by an appropriate definition of the discriminant function.

4.3.1 MODELING

Inspired by hidden Markov model (HMM) type of interactions, we propose to define Ψ to include interactions between input features and labels via multiple copies of the input features as well as features that model interactions between nearby label variables. It is perhaps most intuitive to start from the discriminant function

$$\begin{aligned} F(\mathbf{x}, \mathbf{y}; \mathbf{w}) &= \sum_{t=1}^T \sum_{\sigma \in \Sigma} \langle \bar{\mathbf{w}}_{\sigma}, \Phi(\mathbf{x}^t) \rangle \delta(y^t, \sigma) + \eta \sum_{t=1}^{T-1} \sum_{\sigma \in \Sigma} \sum_{\bar{\sigma} \in \Sigma} \hat{\mathbf{w}}_{\sigma, \bar{\sigma}} \delta(y^t, \sigma) \delta(y^{t+1}, \bar{\sigma}) \\ &= \left\langle \bar{\mathbf{w}}, \sum_{t=1}^T \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t) \right\rangle + \eta \left\langle \hat{\mathbf{w}}, \sum_{t=1}^{T-1} \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1}) \right\rangle. \end{aligned} \quad (13)$$

Here $\mathbf{w} = (\bar{\mathbf{w}}', \hat{\mathbf{w}})'$, Λ^c denotes the orthogonal representation of labels over Σ , and $\eta \geq 0$ is a scaling factor which balances the two types of contributions. It is straightforward to read off the joint feature

map implicit in the definition of the HMM discriminant from Equation (13),

$$\Psi(\mathbf{x}, \mathbf{y}) = \left(\frac{\sum_{t=1}^T \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t)}{\eta \sum_{t=1}^{T-1} \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})} \right).$$

Notice that similar to the multiclass case, we can apply Proposition 21 in the case of an implicit representation of Φ via a kernel function K and the inner product between labeled sequences can thus be written as

$$\langle \Psi((\mathbf{x}, \mathbf{y})), \Psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \rangle = \sum_{s,t=1}^T \delta(y^t, \bar{y}^s) K(\mathbf{x}^t, \bar{\mathbf{x}}^s) + \eta^2 \sum_{s,t=1}^{T-1} \delta(y^t, \bar{y}^s) \delta(y^{t+1}, \bar{y}^{s+1}). \quad (14)$$

A larger family of discriminant functions can be obtained by using more powerful feature functions Ψ . We would like to mention three ways of extending the previous HMM discriminant. First of all, one can extract features not just from \mathbf{x}^t , but from a window around \mathbf{x}^t , e.g. replacing $\Phi(\mathbf{x}^t)$ with $\Phi(\mathbf{x}^{t-r}, \dots, \mathbf{x}^t, \dots, \mathbf{x}^{t+r})$. Since the same input pattern \mathbf{x}^t now occurs in multiple terms, this has been called the use of *overlapping* features (Lafferty et al., 2001) in the context of label sequence learning. Secondly, it is also straightforward to include higher order label-label interactions beyond pairwise interactions by including higher order tensor terms, for instance, label triplets $\sum_t \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1}) \otimes \Lambda^c(y^{t+2})$, etc. Thirdly, one can also combine higher order \mathbf{y} features with input features, for example, by including terms of the type $\sum_t \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})$.

4.3.2 ALGORITHMS

The maximization of $\langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$ over \mathbf{y} can be carried out by dynamic programming, since the cost contributions are additive over sites and contain only linear and nearest neighbor quadratic contributions. In particular, in order to find the best label sequence $\hat{\mathbf{y}} \neq \mathbf{y}_i$, one can perform Viterbi decoding (Forney Jr., 1973; Schwarz and Chow, 1990), which can also determine the second best sequence for the zero-one loss (2-best Viterbi decoding). Viterbi decoding can also be used with other loss functions by computing the maximization for all possible values of the loss function.

4.3.3 SPARSENESS

Proposition 25 Define $R_i \equiv \max_t \|\Phi(\mathbf{x}_i^t)\|$; then $\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}')\|^2 \leq 2T^2(R_i^2 + \eta^2)$.

Proof Notice that $\|\Psi(\mathbf{x}_i, \mathbf{y})\|^2 = \|\sum_t \Phi(\mathbf{x}_i^t) \otimes \Lambda^c(y^t)\|^2 + \eta^2 \|\sum_t \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})\|^2$. The first squared norm can be upper bounded by

$$\left\| \sum_t \Phi(\mathbf{x}_i^t) \otimes \Lambda^c(y^t) \right\|^2 = \sum_s \sum_t \langle \Phi(\mathbf{x}_i^s), \Phi(\mathbf{x}_i^t) \rangle \delta(y^s, y^t) \leq T^2 R_i^2$$

and the second one by $\eta^2 T^2$, which yields the claim. ■

4.4 Sequence Alignment

Next we show how to apply the proposed algorithm to the problem of learning to align sequences $\mathbf{x} \in \Sigma^*$, where Σ^* is the set of all strings over some finite alphabet Σ . For a given pair of sequences

$\mathbf{x} \in \Sigma^*$ and $\mathbf{y} \in \Sigma^*$, alignment methods like the Smith-Waterman algorithm select the sequence of operations (e.g. insertion, substitution) that transforms \mathbf{x} into \mathbf{y} and that maximizes a linear objective function

$$\hat{\mathbf{a}}(\mathbf{x}, \mathbf{y}) = \operatorname{argmax}_{\mathbf{a} \in \mathcal{A}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}, \mathbf{a}) \rangle$$

that is parameterized by the operation scores \mathbf{w} . $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{a})$ is the histogram of alignment operations. The value of $\langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}, \hat{\mathbf{a}}(\mathbf{x}, \mathbf{y})) \rangle$ can be used as a measure of similarity between \mathbf{x} and \mathbf{y} . It is the score of the highest scoring sequence of operations that transforms \mathbf{x} into \mathbf{y} . Such alignment models are used, for example, to measure the similarity of two protein sequences.

4.4.1 MODELING

In order to learn the score vector \mathbf{w} we use training data of the following type. For each native sequence \mathbf{x}_i there is a most similar homologous sequence \mathbf{y}_i along with the optimal alignment \mathbf{a}_i . In addition we are given a set of decoy sequences \mathbf{y}_i^t , $t = 1, \dots, k$ with unknown alignments. Note that this data is more restrictive than what Ristad and Yianilos (1997) consider in their generative modeling approach. The goal is to learn a discriminant function f that recognizes the homologous sequence among the decoys. In our approach, this corresponds to finding a weight vector \mathbf{w} so that homologous sequences align to their native sequence with high score, and that the alignment scores for the decoy sequences are lower. With $\mathcal{Y}_i = \{\mathbf{y}_i, \mathbf{y}_i^1, \dots, \mathbf{y}_i^k\}$ as the output space for the i -th example, we seek a \mathbf{w} so that $\langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{a}_i) \rangle$ exceeds $\langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i^t, \mathbf{a}) \rangle$ for all t and \mathbf{a} . This implies a zero-one loss and hypotheses of the form

$$f(\mathbf{x}_i) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}_i} \max_{\mathbf{a}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}, \mathbf{a}) \rangle. \quad (15)$$

The design of the feature map Ψ depends on the set of operations used in the sequence alignment algorithm.

4.4.2 ALGORITHMS

In order to find the optimal alignment between a given native sequence \mathbf{x} and a homologous/decoy sequence \mathbf{y} as the solution of

$$\max_{\mathbf{a}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}, \mathbf{a}) \rangle, \quad (16)$$

we can use dynamic programming as e.g. in the Smith-Waterman algorithm. To solve the *argmax* in Equation (15), we assume that the number k of decoy sequences is small enough, so that we can select among the scores computed in Equation (16) via exhaustive search.

4.4.3 SPARSENESS

If we select insertion, deletion, and substitution as our possible operations, each (non-redundant) operation reads at least one character in either \mathbf{x} or \mathbf{y} . If the maximum sequence length is N , then the L_1 -norm of $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{a})$ is at most $2N$ and the L_2 -norm of $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{a}) - \Psi(\mathbf{x}, \mathbf{y}', \mathbf{a}')$ is at most $2\sqrt{2}N$.

4.5 Weighted Context-Free Grammars

In natural language parsing, the task is to predict a labeled tree \mathbf{y} based on a string $\mathbf{x} = (x_1, \dots, x_k)$ of terminal symbols. For this problem, our approach extends the approaches of Collins (2000) and Collins and Duffy (2002b) to an efficient maximum-margin algorithm with general loss functions. We assume that each node in the tree corresponds to the application of a context-free grammar rule. The leaves of the tree are the symbols in \mathbf{x} , while interior nodes correspond to non-terminal symbols from a given alphabet \mathcal{N} . For simplicity, we assume that the trees are in Chomsky normal form. This means that each internal node has exactly two children. An exception are pre-terminal nodes (non-leaf nodes that have a terminal symbol as child) which have exactly one child.

4.5.1 MODELING

We consider weighted context-free grammars to model the dependency between \mathbf{x} and \mathbf{y} . Grammar rules are of the form $n_l[C_i \rightarrow C_j, C_k]$ or $n_l[C_i \rightarrow x_t]$, where $C_i, C_j, C_k \in \mathcal{N}$ are non-terminal symbols, and $x_t \in \mathcal{T}$ is a terminal symbol. Each such rule is parameterized by an individual weight w_l . A particular kind of weighted context-free grammar are probabilistic context-free grammars (PCFGs), where this weight w_l is the log-probability of expanding node H_i with rule n_l . In PCFGs, the individual node probabilities are assumed to be independent, so that the probability $P(\mathbf{x}, \mathbf{y})$ of sequence \mathbf{x} and tree \mathbf{y} is the product of the node probabilities in the tree. The most likely parse tree to yield \mathbf{x} from a designated start symbol is the predicted label $h(\mathbf{x})$. This leads to the following maximization problem, where we use $rules(\mathbf{y})$ to denote the multi-set of nodes in \mathbf{y} ,

$$h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \left\{ \sum_{n_l \in rules(\mathbf{y})} w_l \right\}.$$

More generally, weighted context-free grammars can be used in our framework as follows. $\Psi(\mathbf{x}, \mathbf{y})$ contains one feature f_{ijk} for each node of type $n_{ijk}[C_i \rightarrow C_j, C_k]$ and one feature f_{it} for each node of type $n_{it}[C_i \rightarrow x_t]$. As illustrated in Figure 1, the number of times a particular rule occurs in the tree is the value of the feature. The weight vector \mathbf{w} contains the corresponding weights so that $\langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle = \sum_{n_l \in rules(\mathbf{y})} w_l$.

Note that our framework also allows more complex $\Psi(\mathbf{x}, \mathbf{y})$, making it more flexible than PCFGs. In particular, each node weight can be a (kernelized) linear function of the full \mathbf{x} and the span of the subtree.

4.5.2 ALGORITHMS

The solution of $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$ for a given \mathbf{x} can be determined efficiently using a CKY-Parser (see Manning and Schuetze, 1999), which can also return the second best parse for learning with the zero-one loss. To implement other loss functions, like $\Delta(\mathbf{y}_i, \mathbf{y}) = (1 - F_1(\mathbf{y}_i, \mathbf{y}))$, the CKY algorithm can be extended to compute both $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} (1 - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle) \Delta(\mathbf{y}_i, \mathbf{y})$ as well as $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} (\Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle)$ by stratifying the maximization over all values of $\Delta(\mathbf{y}_i, \mathbf{y})$ as described in Joachims (2005) for the case of multivariate classification.

	flt 0/1	tax 0/1	flt Δ	tax Δ	
<i>4 training instances per class</i>					
acc	28.32	28.32	27.47	29.74	+5.01 %
Δ -loss	1.36	1.32	1.30	1.21	+12.40 %
<i>2 training instances per class</i>					
acc	20.20	20.46	20.20	21.73	+7.57 %
Δ -loss	1.54	1.51	1.39	1.33	+13.67 %

Table 1: Results on the WIPO-alpha corpus, section D with 160 groups using 3-fold and 5-fold cross validation, respectively. ‘flt’ is a standard (flat) SVM multiclass model, ‘tax’ the hierarchical architecture. ‘0/1’ denotes training based on the classification loss, ‘ Δ ’ refers to training based on the tree loss.

4.5.3 SPARSENESS

Since the trees branch for each internal node, a tree over a sequence \mathbf{x} of length N has $N - 1$ internal nodes. Furthermore, it has N pre-terminal nodes. This means that the L_1 -norm of $\Psi(\mathbf{x}, \mathbf{y})$ is $2N - 1$ and that the L_2 -norm of $\Psi(\mathbf{x}, \mathbf{y}) - \Psi(\mathbf{x}, \mathbf{y}')$ is at most $\sqrt{4N^2 + 4(N - 1)^2} < 2\sqrt{2}N$.

5. Experimental Results

To demonstrate the effectiveness and versatility of our approach, we applied it to the problems of taxonomic text classification (see also Cai and Hofmann, 2004), named entity recognition, sequence alignment, and natural language parsing.

5.1 Classification with Taxonomies

We have performed experiments using a document collection released by the World Intellectual Property Organization (WIPO), which uses the International Patent Classification (IPC) scheme. We have restricted ourselves to one of the 8 sections, namely section D, consisting of 1,710 documents in the WIPO-alpha collection. For our experiments, we have indexed the title and claim tags. We have furthermore sub-sampled the training data to investigate the effect of the training set size. Document parsing, tokenization and term normalization have been performed with the MindServer retrieval engine.² As a suitable loss function Δ , we have used a tree loss function which defines the loss between two classes \mathbf{y} and \mathbf{y}' as the height of the first common ancestor of \mathbf{y} and \mathbf{y}' in the taxonomy. The results are summarized in Table 1 and show that the proposed hierarchical SVM learning architecture improves performance over the standard multiclass SVM in terms of classification accuracy as well as in terms of the tree loss.

5.2 Label Sequence Learning

We study our algorithm for label sequence learning on a named entity recognition (NER) problem. More specifically, we consider a sub-corpus consisting of 300 sentences from the Spanish news wire article corpus which was provided for the special session of CoNLL2002 devoted to NER.

². This software is available at <http://www.recommind.com>.

Method	HMM	CRF	Perceptron	SVM
Error	9.36	5.17	5.94	5.08

Table 2: Results of various algorithms on the named entity recognition task.

Method	Train Err	Test Err	Const	Avg Loss
SVM ₂	0.2±0.1	5.1±0.6	2824±106	1.02±0.01
SVM ₂ ^{Δ_s}	0.4±0.4	5.1±0.8	2626±225	1.10±0.08
SVM ₂ ^{Δ_m}	0.3±0.2	5.1±0.7	2628±119	1.17±0.12

Table 3: Results for various SVM formulations on the named entity recognition task ($\epsilon = 0.01$, $C = 1$).

The label set in this corpus consists of non-name and the beginning and continuation of person names, organizations, locations and miscellaneous names, resulting in a total of $|\Sigma| = 9$ different labels. In the setup followed in Altun et al. (2003), the joint feature map $\Psi(\mathbf{x}, \mathbf{y})$ is the histogram of state transition plus a set of features describing the emissions. An adapted version of the Viterbi algorithm is used to solve the *argmax* in line 6. For both perceptron and SVM a second degree polynomial kernel was used.

The results given in Table 2 for the zero-one loss, compare the generative HMM with conditional random fields (CRF) (Lafferty et al., 2001), Collins’ perceptron and the SVM algorithm. All discriminative learning methods substantially outperform the standard HMM. In addition, the SVM performs slightly better than the perceptron and CRFs, demonstrating the benefit of a large margin approach. Table 3 shows that all SVM formulations perform comparably, attributed to the fact the vast majority of the support label sequences end up having Hamming distance 1 to the correct label sequence. Notice that for 0-1 loss functions all three SVM formulations are equivalent.

5.3 Sequence Alignment

To analyze the behavior of the algorithm for sequence alignment, we constructed a synthetic dataset according to the following sequence and local alignment model. The native sequence and the decoys are generated by drawing randomly from a 20 letter alphabet $\Sigma = \{1, \dots, 20\}$ so that letter $c \in \Sigma$ has probability $c/210$. Each sequence has length 50, and there are 10 decoys per native sequence. To generate the homologous sequence, we generate an alignment string of length 30 consisting of 4 characters “match”, “substitute”, “insert”, “delete”. For simplicity of illustration, substitutions are always $c \rightarrow (c \bmod 20) + 1$. In the following experiments, matches occur with probability 0.2, substitutions with 0.4, insertion with 0.2, deletion with 0.2. The homologous sequence is created by applying the alignment string to a randomly selected substring of the native. The shortening of the sequences through insertions and deletions is padded by additional random characters.

We model this problem using local sequence alignment with the Smith-Waterman algorithm. Table 4 shows the test error rates (i.e. the percentage of times a decoy is selected instead of the homologous sequence) depending on the number of training examples. The results are averaged over 10 train/test samples. The model contains 400 parameters in the substitution matrix Π and a cost δ for “insert/delete”. We train this model using the SVM₂ and compare against a generative

n	Train Error		Test Error	
	GenMod	SVM ₂	GenMod	SVM ₂
1	20.0±13.3	0.0±0.0	74.3±2.7	47.0±4.6
2	20.0±8.2	0.0±0.0	54.5±3.3	34.3±4.3
4	10.0±5.5	2.0±2.0	28.0±2.3	14.4±1.4
10	2.0±1.3	0.0±0.0	10.2±0.7	7.1±1.6
20	2.5±0.8	1.0±0.7	3.4±0.7	5.2±0.5
40	2.0±1.0	1.0±0.4	2.3±0.5	3.0±0.3
80	2.8±0.5	2.0±0.5	1.9±0.4	2.8±0.6

Table 4: Error rates and number of constraints $|S|$ depending on the number of training examples ($\epsilon = 0.1, C = 0.01$).

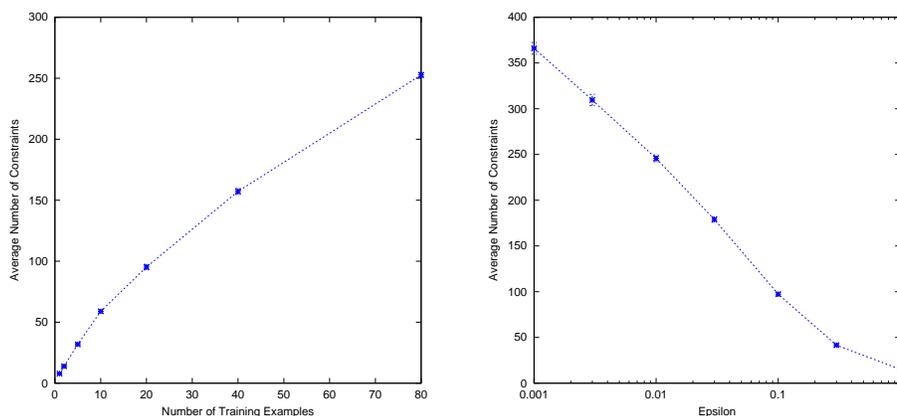


Figure 2: Number of constraints added to S depending on the number of training examples (middle) and the value of ϵ (right). If not stated otherwise, $\epsilon = 0.1, C = 0.01$, and $n = 20$.

sequence alignment model, where the substitution matrix is computed as $\Pi_{ij} = \log\left(\frac{P(x_i, z_j)}{P(x_i)P(z_j)}\right)$ (see e.g. Durbin et al., 1998) using Laplace estimates. For the generative model, we report the results for $\delta = -0.2$, which performs best on the test set. Despite this unfair advantage, the SVM performs better for low training set sizes. For larger training sets, both methods perform similarly, with a small preference for the generative model. However, an advantage of the SVM approach is that it is straightforward to train gap penalties.

Figure 2 shows the number of constraints that are added to S before convergence. The graph on the left-hand side shows the scaling with the number of training examples. As predicted by Theorem 18, the number of constraints is low. It appears to grow sub-linearly with the number of examples. The graph on the right-hand side shows how the number of constraints in the final S changes with $\log(\epsilon)$. The observed scaling appears to be better than suggested by the upper bound in Theorem 18. A good value for ϵ is 0.1. We observed that larger values lead to worse prediction accuracy, while smaller values decrease efficiency while not providing further benefit.

Method	Train				Test				Training Efficiency			
	Acc	Prec	Rec	F_1	Acc	Prec	Rec	F_1	CPU-h	%SVM	Iter	Const
PCFG	61.4	92.4	88.5	90.4	55.2	86.8	85.2	86.0	0	N/A	N/A	N/A
SVM ₂	66.3	92.8	91.2	92.0	58.9	85.3	87.2	86.2	1.2	81.6	17	7494
SVM ₂ ^{Δ_s}	62.2	93.9	90.4	92.1	58.9	88.9	88.1	88.5	3.4	10.5	12	8043
SVM ₂ ^{Δ_m}	63.5	93.9	90.8	92.3	58.3	88.7	88.1	88.4	3.5	18.0	16	7117

Table 5: Results for learning a weighted context-free grammar on the Penn Treebank.

5.4 Weighted Context-Free Grammars

We test the feasibility of our approach for learning a weighted context-free grammar (see Figure 1) on a subset of the Penn Treebank Wall Street Journal corpus. We consider the 4098 sentences of length at most 10 from sections F2-21 as the training set, and the 163 sentences of length at most 10 from F22 as the test set. Following the setup in Johnson (1998), we start based on the part-of-speech tags and learn a weighted grammar consisting of all rules that occur in the training data. To solve the $argmax$ in line 6 of the algorithm, we use a modified version of the CKY parser of Mark Johnson.³

The results are given in Table 5. They show micro-averaged precision, recall, and F_1 for the training and the test set. The first line shows the performance of the generative PCFG model using the maximum likelihood estimate (MLE) as computed by Johnson’s implementation. The second line show the SVM₂ with zero-one loss, while the following lines give the results for the F_1 -loss $\Delta(\mathbf{y}_i, \mathbf{y}) = (1 - F_1(\mathbf{y}_i, \mathbf{y}))$ using SVM₂ ^{Δ_s} and SVM₂ ^{Δ_m} . All results are for $C = 1$ and $\varepsilon = 0.01$. All values of C between 10^{-1} to 10^2 gave comparable prediction performance. While the zero-one loss—which is also implicitly used in Perceptrons (Collins and Duffy, 2002a; Collins, 2002)—achieves better accuracy (i.e. predicting the complete tree correctly), the F_1 -score is only marginally better compared to the PCFG model. However, optimizing the SVM for the F_1 -loss gives substantially better F_1 -scores, outperforming the PCFG substantially. The difference is significant according to a McNemar test on the F_1 -scores. We conjecture that we can achieve further gains by incorporating more complex features into the grammar, which would be impossible or at best awkward to use in a generative PCFG model. Note that our approach can handle arbitrary models (e.g. with kernels and overlapping features) for which the $argmax$ in line 6 can be computed. Experiments with such complex features were independently conducted by Taskar et al. (2004b) based on the algorithm in Taskar et al. (2004a). While their algorithm cannot optimize F1-score as the training loss, they report substantial gains from the use of complex features.

In terms of training time, Table 5 shows that the total number of constraints added to the working set is small. It is roughly twice the number of training examples in all cases. While the training is faster for the zero-one loss, the time for solving the QPs remains roughly comparable. The re-scaling formulations lose time mostly on the $argmax$ in line 6 of the algorithm. This might be sped up, since we were using a rather naive algorithm in the experiments.

6. Conclusions

We presented a maximum-margin approach to learning functional dependencies for complex output spaces. In particular, we considered cases where the prediction is a structured object or where the prediction consists of multiple dependent variables. The key idea is to model the problem as

3. This software is available at <http://www.cog.brown.edu/~mj/Software.htm>.

a (kernelized) linear discriminant function over a joint feature space of inputs and outputs. We demonstrated that our approach is very general, covering problems from natural language parsing and label sequence learning to multilabel classification and classification with output features.

While the resulting learning problem can be exponential in size, we presented an algorithm for which we prove polynomial convergence for a large class of problems. We also evaluated the algorithm empirically on a broad range of applications. The experiments show that the algorithm is feasible in practice and that it produces promising results in comparison to conventional generative models. A key advantage of the algorithm is the flexibility to include different loss functions, making it possible to directly optimize the desired performance criterion. Furthermore, the ability to include kernels opens the opportunity to learn more complex dependencies compared to conventional, mostly linear models.

References

- Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.
- L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the ACM Thirteenth Conference on Information and Knowledge Management*, 2004.
- W. Cohen, R. Shapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.
- M. Collins. Discriminative reranking for natural language parsing. In *Proceedings of the Seventieth International Conference on Machine Learning*, 2000.
- M. Collins. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.
- M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632, 2002a.
- M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the Fortieth Annual Meeting of the Association for Computational Linguistics*, 2002b.
- K. Crammer and Y. Singer. On the algorithmic implementation of multi-class kernel-based vector machines. *Machine Learning Research*, 2:265–292, 2001.
- K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, 2002.
- R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.
- G. D. Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, 1973.

- M. Grötschel, L. Lovàt, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
- S. Har-Peled, D. Roth, and D. Zimak. Constraint classification for multiclass classification and ranking. In *Advances in Neural Information Processing Systems 14*, 2002.
- R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.
- T. Hofmann, I. Tsochantaridis, and Y. Altun. Learning over structured output spaces via joint kernel functions. In *Proceedings of the Sixth Kernel Workshop*, 2002.
- T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2002.
- T. Joachims. Learning to align sequences: A maximum-margin approach. Technical report, Cornell University, 2003.
- T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 2005.
- M. Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4): 613–632, 1998.
- N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4): 373–396, 1984.
- J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, 8:703–712, 1960.
- J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.
- C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.
- E. Ristad and P. Yianilos. Learning string edit distance. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 287–295, 1997.
- R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168, 2000.
- R. Schwarz and Y. L. Chow. The n-best algorithm: An efficient and exact procedure for finding the n most likely hypotheses. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 81–84, 1990.
- A. Smola and T. Hofmann. Exponential families for generative and discriminative estimation. Technical report, 2003.

- B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*, 2004a.
- B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-Margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004b.
- I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.
- V. Vapnik. *Statistical Learning Theory*. Wiley and Sons Inc., New York, 1998.
- J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *Advances in Neural Information Processing Systems 15*, 2003.
- J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.
- D. H. Younger. Recognition and parsing of context-free languages in time n^3 . *Information and Control*, 2(10):189–208, 1967.

Frames, Reproducing Kernels, Regularization and Learning

Alain Rakotomamonjy

ALAIN.RAKOTOMAMONJY@INSA-ROUEN.FR

Stéphane Canu

STEPHANE.CANU@INSA-ROUEN.FR

Perception, Systèmes et Information CNRS FRE2645

INSA de Rouen

76801 Saint Etienne du Rouvray, France

Editor: Alex Smola

Abstract

This work deals with a method for building a reproducing kernel Hilbert space (RKHS) from a Hilbert space with frame elements having special properties. Conditions on existence and a method of construction are given. Then, these RKHS are used within the framework of regularization theory for function approximation. Implications on semiparametric estimation are discussed and a multiscale scheme of regularization is also proposed. Results on toy and real-world approximation problems illustrate the effectiveness of such methods.

Keywords: regularization, kernel, frames, wavelets

1. Introduction

A reproducing kernel Hilbert space (RKHS) is a Hilbert space of functions with special properties (Aronszajn, 1950). It plays an important role in approximation and regularization theory as it allows writing in a simple way the solution of a learning from empirical data problem (Wahba, 1990, 2000). Since the development of support vector machines (SVMs) (Vapnik, 1995; Vapnik et al., 1997; Burges, 1998; Vapnik, 1998) as a machine learning for data classification and functional estimation, there is a growing interest around reproducing kernel Hilbert spaces. In fact, for nonlinear classification or approximation, SVMs map the input space into a high dimensional feature space by means of a nonlinear transformation Φ (Boser et al., 1992). Usually in SVMs, the mapping function is related to an integral operator kernel $K(x, y)$ which corresponds to the dot product of the mapped data:

$$K(x, y) = \langle \Phi(x), \Phi(y) \rangle$$

where x and y belong to the input space.

In regularization theory (Tikhonov and Arsénin, 1977; Groetsch, 1993; Morosov, 1984), the ill-conditioned estimation from data problem is transformed into a well-conditioned problem by means of a stabilizer, which is a functional with specific properties.

For both SVMs and regularization theory, one can consider special cases of kernel and stabilizer: the kernel and the norm associated with an RKHS (Girosi, 1998; Smola et al., 1998; Evgeniou et al., 2000). This justifies the appeal of RKHS as it allows the development of a general framework that includes several approximation schemes.

One of the most important issues in a learning problem is the choice of the data representation. For instance, in SVMs this corresponds to the selection of the nonlinear mapping Φ . It is a key

problem since the mapping has a direct influence on the kernel and thus, it has an influence on the solution of the approximation or classification problem. In practical cases, the choice of an appropriate data representation is as important as the choice of the learning machine. In fact, prior information on a specific problem can be used for choosing an efficient input representation, or for choosing a good hypothesis space that leads to enhanced performance of the learning machine (Scholkopf et al., 1998; Jaakkola and Haussler, 1999; Niyogi et al., 1998).

The purpose of this paper is to present a method for constructing an RKHS and its associated kernel by means of frame theory (Duffin and Schaeffer, 1952; Daubechies, 1992). A frame of a Hilbert space spans any vector of the space by linear combination of the frame elements. But unlike a basis, a frame is not necessarily linear independent although it achieves stable representation. Since a frame is a more general way to represent elements of Hilbert space, it allows flexibility in the representation of any vector of the space. By giving conditions for constructing arbitrary RKHS from frame elements, our goal is to widen the choice of kernel so that in future applications, one can adapt its RKHS to prior information available concerning a problem at hand.

The paper is organized as follows: in Section 2, we recall the problem of estimating function from data and the way of solving it owing to regularization theory. Section 3 deals with frame. After a short introduction about frame theory, we give conditions for a Hilbert space described by a frame to be an RKHS and then derive the corresponding kernel. In Section 4, a practical way for building RKHS is given. Section 5 discusses implication of these results on regularization technique and proposes an algorithm for multiscale approximation. Section 6 presents estimation results on numerical experiments on toy and real-world problems while Section 7 concludes the paper and contains remarks and other issues about this work.

2. Regularized Approximation

As argued by Girosi et al. (1995), learning from data can be viewed as a multivariate function approximation from sparse data. Supposing that one has a set of data $\{(x_i, y_i), x_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1 \dots \ell\}$ provided by the random sampling of a noisy function f , the goal is to recover the unknown function f , from the knowledge of the data set. It is well-known that such a problem is ill-posed as there exists an infinity of functions that pass perfectly through the data. One way to transform this problem into a well-posed one is to assume that the function f presents some smoothness properties and hence, the problem becomes a variational problem of finding the function f^* that minimizes the functional (Tikhonov and Arsénin, 1977):

$$H[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} C(y_i, f(x_i)) + \lambda \Omega[f] \tag{1}$$

where λ is a positive number, C a cost function which determines how differences between $f(x_i)$ and y_i should be penalized and $\Omega[f]$ a functional which denotes the prior information on the function f . λ balances the trade-off between fitness of f to the data and smoothness of f . This regularization principle leads to different approximation schemes depending on the cost function $C(\cdot, \cdot)$. Classical L_2 cost function ($C(y_i), f(x_i) = (y_i - f(x_i))^2$) leads to the so-called Regularization Networks (Girosi et al., 1995; Evgeniou et al., 2000) whereas cost function like Vapnik's ϵ -insensitive function leads to SVMs.

When the functional $\Omega[f]$ is defined as $\|f\|_{\mathcal{H}}^2$, the square norm of f in a reproducing kernel Hilbert space \mathcal{H} associated to a positive definite function K (the square norm in a Hilbert space

being related to the inner product by $\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}}$, the solution of Equation (1) is under general conditions

$$f^*(x) = \sum_{i=1}^{\ell} c_i K(x, x_i). \quad (2)$$

The case of $\|f\|_{\mathcal{H}}$ being a seminorm leads to a minimizer with the following form:

$$f^*(x) = \sum_{i=1}^{\ell} c_i K(x, x_i) + \sum_{j=1}^m d_j g_j(x) \quad (3)$$

where $\{g_j\}_{j=1\dots m}$ span the null space of the functional $\|f\|_{\mathcal{H}}^2$.

In a nutshell, looking for a function f of the form (3) is equivalent to minimizing the functional $H[f]$, and thus the solution which depends on λ is the “best” balance between smoothness in \mathcal{H} and fitness to the data. Choosing a kernel K is equivalent to specifying a prior information on the RKHS, therefore having a large choice of RKHS should be fruitful for the approximation accuracy, if overfitting is properly controlled, since one can adapt its hypothesis space to each specific data set.

3. Frames and Reproducing Kernel Hilbert Spaces

In this section, we give an introduction to frame theory that will be useful for the remainder of the paper.

3.1 A Brief Review of Frame Theory

Frame theory was introduced by Duffin and Schaeffer (1952) (Daubechies, 1992) in order to establish general conditions under which one can reconstruct perfectly a function f in a Hilbert space \mathcal{H} from its inner product $(\langle \cdot, \cdot \rangle_{\mathcal{H}})$ with a family of vectors $\{\phi_n\}_{n \in \Gamma}$ with Γ being a finite or infinite countable index set.

Definition 1 *A set of vectors $\{\phi_n\}_{n \in \Gamma}$ is a frame of a Hilbert space \mathcal{H} if there exists two constants $A > 0$ and $\infty > B \geq A > 0$ so that*

$$\forall f \in \mathcal{H}, \quad A \|f\|_{\mathcal{H}}^2 \leq \sum_{n \in \Gamma} |\langle f, \phi_n \rangle_{\mathcal{H}}|^2 \leq B \|f\|_{\mathcal{H}}^2. \quad (4)$$

The frame is said to be tight if A and B are equal.

If the set $\{\phi_n\}_{n \in \Gamma}$ satisfies the frame condition then the frame operator U can be defined as

$$U : \begin{array}{l} \mathcal{H} \longrightarrow \ell^2 \\ f \longrightarrow \{\langle f, \phi_n \rangle_{\mathcal{H}}\}_{n \in \Gamma}. \end{array} \quad (5)$$

The reconstruction of f from its frame coefficients needs the definition of a dual frame. For this purpose, one introduces the adjoint operator U^* of U which exists and is unique because it lies on a Hilbert space:

$$U^* : \begin{array}{l} \ell^2 \longrightarrow \mathcal{H} \\ \{c_n\}_{n \in \Gamma} \longrightarrow \sum_{n \in \Gamma} c_n \phi_n. \end{array} \quad (6)$$

Theorem 1 (Daubechies, 1992) *Let $\{\phi_n\}_{n \in \Gamma}$ be a frame of \mathcal{H} with frame bounds A and B . Let us define the dual frame $\{\bar{\phi}_n\}_{n \in \Gamma}$ as $\bar{\phi}_n = (U^*U)^{-1}\phi_n$. For all $f \in \mathcal{H}$, we have*

$$\frac{1}{B} \|f\|_{\mathcal{H}}^2 \leq \sum_{n \in \Gamma} |\langle f, \bar{\phi}_n \rangle_{\mathcal{H}}|^2 \leq \frac{1}{A} \|f\|_{\mathcal{H}}^2 \tag{7}$$

and

$$f = \sum_{n \in \Gamma} \langle f, \bar{\phi}_n \rangle_{\mathcal{H}} \phi_n = \sum_{n \in \Gamma} \langle f, \phi_n \rangle_{\mathcal{H}} \bar{\phi}_n. \tag{8}$$

If the frame is tight then $\bar{\phi}_n = \frac{1}{A} \phi_n$.

■

This theorem also shows that the dual frame $\{\bar{\phi}_n\}_{n \in \Gamma}$ is a family of vectors which allows to recover any $f \in \mathcal{H}$, and consequently one can write each vector of the frame and the dual frame as

$$\forall m \in \Gamma, \quad \bar{\phi}_m = \sum_{n \in \Gamma} \langle \bar{\phi}_m, \phi_n \rangle_{\mathcal{H}} \bar{\phi}_n \tag{9}$$

and

$$\forall m \in \Gamma, \quad \phi_m = \sum_{n \in \Gamma} \langle \phi_m, \phi_n \rangle_{\mathcal{H}} \bar{\phi}_n. \tag{10}$$

According to this theorem and the above equations, one can note that an orthonormal basis of \mathcal{H} is a special case of frame where $A = B = 1$, $\bar{\phi}_n = \phi_n$ and $\|\phi_n\| = 1$. However, as stated by Daubechies (1992), frame redundancy can be statistically useful. Also note that in the general case, we do not have an analytical expression of the dual frame, and thus it has to be computed numerically. Grochenig has proposed such an algorithm (Grochenig, 1993) which is based on a iterative conjugate gradient method. We have briefly described this algorithm in the appendix but for further details, one should refer to the original paper.

For the sake of simplicity, in the following we will call frameable Hilbert space, a Hilbert space \mathcal{H} for which there exists a set of vector of \mathcal{H} that forms a frame of \mathcal{H} . Note that all separable Hilbert spaces are frameable since by definition they have a countable orthonormal basis.

3.2 A Reproducing Kernel Hilbert Space and Its Frame

After this short introduction on frame theory, let us look at the conditions under which a frameable Hilbert space is also a reproducing kernel Hilbert space.

First of all, we introduce some notations that will be used throughout the rest of the paper: let \mathbb{R}^{Ω} be the set of all functions defined on a domain $\Omega \subset \mathbb{R}^d$ with values in \mathbb{R} .

For the purpose of being self-contained, we propose here some useful definitions and properties concerning RKHS. However, the reader who is interested in deeper details can refer to books describing mathematical aspects (Atteia, 1992; Berlinet and Agnan, 2004).

Definition 2 *A Hilbert space \mathcal{H} with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is a reproducing kernel Hilbert space of \mathbb{R}^{Ω} if:*

- \mathcal{H} is a subspace of \mathbb{R}^{Ω}

- $\forall t \in \Omega$, $\exists M_t > 0$ so that

$$\forall f \in \mathcal{H}, \quad |f(t)| \leq M_t \|f\|. \quad (11)$$

This latter property means that for any $t \in \Omega$, the linear functional \mathcal{F}_t (also called the evaluation functional) defined as

$$\mathcal{F}_t(f): \begin{array}{l} \mathcal{H} \longrightarrow \mathbb{R} \\ f \longrightarrow \mathcal{F}_t(f) = f(t) \end{array}$$

is a bounded linear functional.

Note that for any Hilbert space of functions, the evaluation functional is linear, thus the important point for having the reproducing kernel property is this evaluational functional being bounded.

Definition 3 We call $\text{Hilb}(\mathbb{R}^\Omega)$ the set of all RKHS of \mathbb{R}^Ω .

Owing to the Riesz theorem, one can state that:

Theorem 4 Let $\mathcal{H} \in \text{Hilb}(\mathbb{R}^\Omega)$, there exists an unique symmetric function $K(\cdot, t)$ of \mathcal{H} called the reproducing kernel of \mathcal{H} so that

$$\forall t \in \Omega, \quad \forall f \in \mathcal{H}, \quad f(t) = \langle f | K(\cdot, t) \rangle_{\mathcal{H}}. \quad (12)$$

■

Theorem 5 Let \mathcal{H} be a Hilbert space and $\{\phi_n\}_{n \in \Gamma}$ be a frame of this space. If $\{\phi_n\}_{n \in \Gamma}$ is a (finite or infinite) set of functions of \mathbb{R}^Ω , so that:

$$\forall t \in \Omega, \quad \left\| \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t) \right\|_{\mathcal{H}} < \infty. \quad (13)$$

Then \mathcal{H} is a reproducing kernel Hilbert space.

■

Proof

Step 1 Any ϕ_n is both an element of \mathbb{R}^Ω and \mathcal{H} . Hence the equation

$$\forall f \in \mathcal{H}, \quad f = \sum_{n \in \Gamma} \langle f, \bar{\phi}_n \rangle_{\mathcal{H}} \phi_n$$

holds in \mathcal{H} according to the frame property given in Equation (8) (Mallat, 1998; Daubechies, 1992). Now since, \mathbb{R}^Ω has a structure of vector space, $f = \sum_{n \in \Gamma} \langle f, \bar{\phi}_n \rangle \phi_n$ is also valid in \mathbb{R}^Ω and thus f also belongs to \mathbb{R}^Ω . Now, if for each $t \in \Omega$, we define the seminorm on the vector space \mathbb{R}^Ω as

$$\forall f \in \mathbb{R}^\Omega, \quad \|f\|_t = |f(t)|.$$

According to this seminorm, we get the following pointwise convergence:

$$f = \sum_{n \in \Gamma} \langle f, \bar{\phi}_n \rangle_{\mathcal{H}} \phi_n \Leftrightarrow f(t) = \sum_{n \in \Gamma} \langle f, \bar{\phi}_n \rangle_{\mathcal{H}} \phi_n(t). \quad (14)$$

Step 2 Now let's show that $\forall t \in \Omega, \exists M_t > 0$ so that

$$\forall f \in \mathcal{H}, \quad |f(t)| \leq M_t \|f\|_{\mathcal{H}}. \tag{15}$$

All elements of \mathcal{H} can be expanded with regards to the frame elements, so according to Equation (14), we have for all f in \mathcal{H} and \mathbb{R}^Ω :

$$|f(t)| = \left| \sum_{n \in \Gamma} \langle f(\cdot), \bar{\phi}_n(\cdot) \rangle_{\mathcal{H}} \phi_n(t) \right| \tag{16}$$

and consequently,

$$\begin{aligned} |f(t)| &= \left| \left\langle f(\cdot), \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t) \right\rangle_{\mathcal{H}} \right| \\ &\leq \|f\|_{\mathcal{H}} \left\| \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t) \right\|_{\mathcal{H}} \end{aligned} \tag{17}$$

by defining $M_t \triangleq \left\| \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t) \right\|_{\mathcal{H}}$ one can conclude that \mathcal{H} is a reproducing kernel Hilbert space since M_t is finite by hypothesis and therefore, \mathcal{H} admits an unique reproducing kernel. ■

Remark 6 *In this proof, we have chosen to expand a function f of \mathcal{H} according to $f = \sum_{n \in \Gamma} \langle f, \bar{\phi}_n \rangle \phi_n$. However choosing the relationship $f = \sum_{n \in \Gamma} \langle f, \phi_n \rangle \bar{\phi}_n$ would have led to the following equivalent condition to Equation (13):*

$$\forall t \in \Omega, \quad \left\| \sum_{n \in \Gamma} \bar{\phi}_n(t) \phi_n(\cdot) \right\|_{\mathcal{H}} < \infty. \tag{18}$$

Now let's try to express the reproducing kernel of such a Hilbert space.

Theorem 7 *Let \mathcal{H} be a reproducing kernel Hilbert space and $\mathcal{H} \in \text{Hilb}(\mathbb{R}^\Omega)$, and the family $\{\phi_n\}_{n \in \Gamma}$ be a frame of this space, the reproducing kernel is $K(s, t)$ defined by:*

$$K : \begin{cases} \Omega \times \Omega \rightarrow \mathbb{R} \\ s \times t \rightarrow K(s, t) = \sum_{n \in \Gamma} \bar{\phi}_n(s) \phi_n(t) \end{cases} \tag{19}$$

Proof

At first, note that according to the frame inequality:

$$\sum_{n \in \Gamma} \phi_n^2(t) = \sum_{n \in \Gamma} |\langle K(t, \cdot), \phi_n(\cdot) \rangle_{\mathcal{H}}|^2 \leq B \|K(t, \cdot)\|_{\mathcal{H}}^2 < \infty.$$

Furthermore, according to Theorem (1) we know that $\{\bar{\phi}_n\}_{n \in \Gamma}$ is another frame of \mathcal{H} . Thus, similarly to Equation (6), we can define the adjoint operator $U_{\bar{\phi}}^*$ associated to this dual frame. And,

applying $U_{\bar{\phi}}^*$ to the ℓ_2 sequence $\{\phi_n(t)\}$ shows that the function $\sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t)$ is a well-defined function of \mathcal{H} .

Furthermore, any $f \in \mathcal{H}$ can be expanded by means of the frame of \mathcal{H} , thus according to Equation (14):

$$\begin{aligned} f(t) &= \sum_{n \in \Gamma} \langle f, \bar{\phi}_n \rangle_{\mathcal{H}} \phi_n(t) \\ &= \left\langle f(\cdot), \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t) \right\rangle_{\mathcal{H}} \end{aligned} \quad (20)$$

and since \mathcal{H} is an RKHS, we have

$$\forall f \in \mathcal{H}, \quad \forall t \in \Omega, \quad f(t) = \langle f(\cdot), K(\cdot, t) \rangle_{\mathcal{H}}. \quad (21)$$

Hence, by identifying Equation (20) and (21) due to the unicity of the reproducing kernel, we have

$$K(\cdot, t) = \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t)$$

and thus, we can conclude that

$$K(s, t) = \sum_{n \in \Gamma} \bar{\phi}_n(s) \phi_n(t).$$

■

These propositions show that a Hilbert space which can be described by its frame is under general conditions, a reproducing kernel Hilbert space and its reproducing kernel is given by a linear combination of its frame and dual frame product.

A simple corollary to Theorem (7) is that for any RKHS \mathcal{H} with family $\{\phi_n\}_{n \in \Gamma}$ as a frame, the inequality (13) holds. This naturally stems from the fact that $K(\cdot, t) = \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t)$ is a well-defined function of \mathcal{H} (as stated in the proof of Theorem 7) and thus it has a finite norm in \mathcal{H} .

The symmetry and the positivity of the kernel $K(s, t)$ are direct consequences of $K(\cdot, \cdot)$ being a kernel of an RKHS. However, these properties can also be easily shown owing to the frame representation. In fact, according to Equation (8) and (14), we get:

$$\begin{aligned} x(t) &= \sum_{n \in \Gamma} \langle x, \bar{\phi}_n \rangle_{\mathcal{H}} \phi_n(t) = \sum_{n \in \Gamma} \langle x, \phi_n \rangle_{\mathcal{H}} \bar{\phi}_n(t) \\ &= \left\langle x(\cdot), \sum_{n \in \Gamma} \bar{\phi}_n(\cdot) \phi_n(t) \right\rangle_{\mathcal{H}} = \left\langle x(\cdot), \sum_{n \in \Gamma} \phi_n(\cdot) \bar{\phi}_n(t) \right\rangle_{\mathcal{H}} \end{aligned} \quad (22)$$

thus, owing to the uniqueness of the functional evaluation in a RKHS, one can deduce from Equation (22) that

$$K(s, t) = \sum_{n \in \Gamma} \bar{\phi}_n(s) \phi_n(t) = \sum_{n \in \Gamma} \bar{\phi}_n(t) \phi_n(s) = K(t, s).$$

The positivity can also be proved from the following reasoning. Let x_1, \dots, x_ℓ be some vectors of Ω and a_1, \dots, a_ℓ some scalar values in \mathbb{R} , we want to show that for any set $\{x_i\}$ and $\{a_i\}$:

$$\sum_{i,j}^{\ell} a_i a_j K(x_i, x_j) \geq 0.$$

According to Equation (10), we can write

$$K(x_i, x_j) = \sum_{n \in \Gamma} \bar{\phi}_n(x_i) \sum_{m \in \Gamma} \bar{\phi}_m(x_j) \langle \phi_n, \phi_m \rangle_{\mathcal{H}}.$$

Thus, we have

$$\begin{aligned} \sum_{i,j}^{\ell} a_i a_j K(x_i, x_j) &= \sum_{i,j}^{\ell} a_i a_j \sum_{n,m \in \Gamma} \bar{\phi}_n(x_i) \bar{\phi}_m(x_j) \langle \phi_n, \phi_m \rangle_{\mathcal{H}} \\ &= \left\langle \sum_i^{\ell} \sum_{n \in \Gamma} a_i \bar{\phi}_n(x_i) \phi_n(\cdot), \sum_j^{\ell} \sum_{m \in \Gamma} a_j \bar{\phi}_m(x_j) \phi_m(\cdot) \right\rangle_{\mathcal{H}} \\ &= \left\| \sum_i^{\ell} \sum_{n \in \Gamma} a_i \bar{\phi}_n(x_i) \phi_n(\cdot) \right\|_{\mathcal{H}}^2 \\ &\geq 0. \end{aligned}$$

4. Learning Schemes Using Frames

In the previous section, conditions for a frameable Hilbert space being an RKHS were given. Here, we are interested in constructing a reproducing kernel Hilbert space together with its frame and discuss about the implications of such result in a functional estimation framework.

4.1 Learning on Frameable Hilbert Spaces

An interesting point of frameable Hilbert space is that under weak conditions, it becomes easy to build RKHS. The following theorem proves such point.

Theorem 8 *Let $N \in \mathbb{N}$ and $\{\phi_n\}_{n=1 \dots N}$ be a finite set of non-zero functions of a Hilbert space $(\mathcal{B}, \langle \cdot, \cdot \rangle)$ with $\mathcal{B} \subset \mathbb{R}^{\Omega}$ so that*

$$\exists M, \forall t \in \Omega, \forall n \ 1 \leq n \leq N, \quad |\phi_n(t)| \leq M.$$

Let \mathcal{H} be the set of functions so that

$$\mathcal{H} = \left\{ f = \sum_{n=1}^N a_n \phi_n : a_n \in \mathbb{R}, \ n = 1, \dots, N \right\}$$

$(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{B}})$ is an RKHS and its reproducing kernel is

$$K(s, t) = \sum_{n=1}^N \bar{\phi}_n(s) \phi_n(t),$$

where $\{\bar{\phi}_n\}_{n=1, \dots, N}$ is the dual frame of $\{\phi_n\}_{n=1, \dots, N}$ in \mathcal{H} .

■

Proof

Step 1 \mathcal{H} is a Hilbert space.

This is straightforward since \mathcal{H} is a closed subspace of a Hilbert space \mathcal{B} , and is endowed with \mathcal{B} inner product. Hence \mathcal{H} is a Hilbert space.

Step 2 $\{\phi_n\}$ is a frame of \mathcal{H} . A proof of this step is also given in Christensen (1993). We have to show that there exists A and B satisfying equation (4). Let us consider the non trivial case that $\text{span}\{\phi_n\}_{n=1..N} \neq 0$.

The existence of B is straightforward applying Cauchy-Schwartz inequality. In fact, for all $f \in \mathcal{H}$

$$|\langle f, \phi_n \rangle|^2 \leq \|f\|^2 \|\phi_n\|^2$$

and thus

$$\sum_{n=1}^N |\langle f, \phi_n \rangle|^2 \leq \|f\|^2 \sum_{n=1}^N \|\phi_n\|^2.$$

Thus by taking $B = \sum_{n=1}^N \|\phi_n\|^2$, we have $B < \infty$ and B satisfies the right-hand inequality of Equation (4).

Let $\mathcal{H}^* \triangleq \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} > 0\}$ and $S(f)$ be the following mapping:

$$S: \begin{cases} \mathcal{H}^* & \longrightarrow \mathbb{R} \\ f & \longrightarrow S(f) = \sum_{n \in \Gamma} |\langle f, \phi_n \rangle|^2. \end{cases} \tag{23}$$

This mapping is continuous and because \mathcal{H}^* is of finite dimension the restriction of S to the unit ball in $\text{span}\{\phi_n\}_{n=1..N}$ reach its infimum (Brezis, 1983): there is $g \in \text{span}\{\phi_n\}_{n=1, \dots, N}$ with $\|g\| = 1$ such that

$$\sum_{n \in \Gamma} |\langle g, \phi_n \rangle|^2 = \inf \left\{ \sum_{n \in \Gamma} |\langle f, \phi_n \rangle|^2, \quad f \in \mathcal{H}^* \text{ so that } \|f\| = 1 \right\}.$$

Let A be $\sum_{n \in \Gamma} |\langle g, \phi_n \rangle|^2$. Hence $A > 0$, and as $\|g\| = 1$, one has for any $f \in \mathcal{H}^*$:

$$A \|f\|^2 \leq \sum_{n=1}^N |\langle f, \phi_n \rangle|^2.$$

Step 3 Now let's prove that \mathcal{H} is an RKHS. For that it suffices to prove that the frame $\{\phi_n\}$ satisfies condition given in Theorem 5.

This is straightforward since $\{\phi_n\}_{n=1, \dots, N}$ is a frame of \mathcal{H} and owing to Theorem 1, the dual frame $\{\bar{\phi}_n\}_{n=1, \dots, N}$ is also a frame of \mathcal{H} . Hence, the norm of each $\bar{\phi}_n$ is finite. Besides, $|\phi_n(t)|$ is supposed to be bounded by M . Hence,

$$\left\| \sum_{n=1}^N \bar{\phi}_n(\cdot) \phi_n(t) \right\| \leq M \sum_{n=1}^N \|\bar{\phi}_n(\cdot)\| < \infty$$

and consequently, \mathcal{H} is an RKHS with a kernel equal to:

$$K(s, t) = \sum_{n=1}^N \bar{\phi}_n(s) \phi_n(t).$$

■

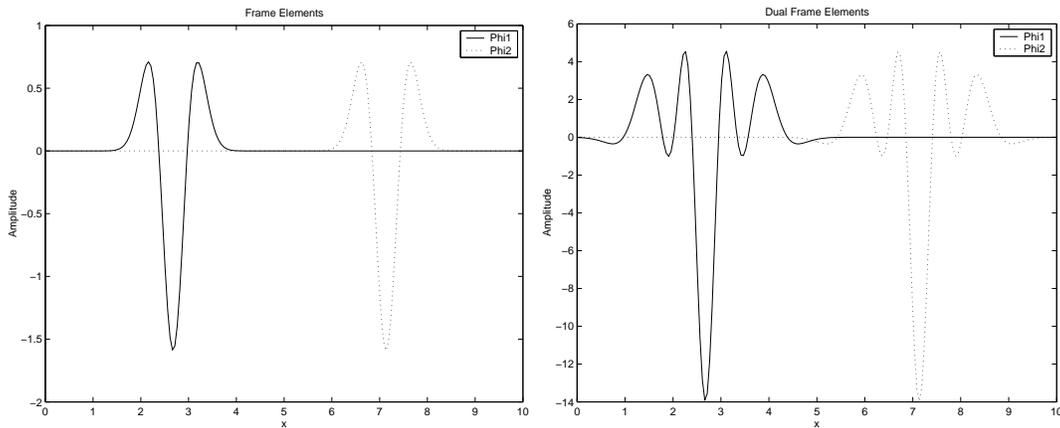


Figure 1: Examples of wavelet frame elements (left) and their dual elements (right).

Here, we give some examples of RKHS that have been derived from the direct application of this theorem.

Example 1 Any finite set of bounded, real-valued, pointwise-defined and square integrable functions on Ω endowed with the inner product $\langle f, g \rangle = \int_{\Omega} f(t)g(t)dt$ spans a RKHS. For instance, the set of functions whose expressions are given below spans an RKHS.

$$\forall t \in \Omega, \phi_n(t) = t \cdot e^{-(t-n)^2}, \quad n \in [n_{min}, n_{max}] \text{ with } (n_{min}, n_{max}) \in \mathbb{N}^2$$

Example 2 Any finite set of bounded and pointwise-defined functions belonging to Sobolev space (Berlinet and Agnan, 2004) spans an RKHS. The set of functions, given in the previous example spans also an RKHS in a Sobolev inner product sense.

Example 3 Consider a finite set of wavelet on \mathbb{R}

$$\left\{ \Psi_{j,k}(t) = \frac{1}{\sqrt{a^j}} \Psi \left(\frac{t - ku_0 a^j}{a^j} \right), j \in \mathbb{Z} : j_{min} \leq j \leq j_{max}, k \in \mathbb{Z} : k_{min} \leq k \leq k_{max} \right\}$$

where $(a, u_0) \in \mathbb{R}_+^* \times \mathbb{R}$, and $(j_{min}, j_{max}, k_{min}, k_{max}) \in \mathbb{Z}^4$. Then the span of these functions endowed with the inner product $\langle f, g \rangle = \int_{\mathbb{R}} f(t)g(t)dt$ is an RKHS. Figure (1) plots an example of wavelet frame and dual frame elements for a dilation $j = -7$.

The main interest of Theorem (8) is the flexibility it introduces in the RKHS choice or in the choice of the functions that span the hypothesis space. However, this theorem only deals with finite dimensional RKHS. For building infinite dimensional RKHS, Theorem (5) has to be used. The main difference between the finite and infinite dimensional case and thus between Theorems (5) and (8) is that a finite set of functions $\{\phi_n\}_{n=1, \dots, N}$, if endowed with an adequate inner product, is always a frame of the space it spans (see step 2 of the proof of Theorem (8)). This is not always true for an infinite set of functions and in this case, the frame condition given in Equation (4) and the boundedness of the evaluation functional in Equation (13) have to be verified. Next examples are examples of infinite dimension RKHS which kernels are given explicitly by their frame elements.

Example 4 Let us consider \mathcal{H} as the space of continuous and differentiable functions on $\Omega = [0, 1]$ with the constraints that for any $f \in \mathcal{H}$, $f(0) = f(1) = 0$ and $\partial f \in L_2(\Omega)$ where ∂f is the usual derivative of f . Endowed with the inner product:

$$\forall f \text{ and } g \in \mathcal{H}, \langle f, g \rangle_{\mathcal{H}} = \int_{\Omega} \partial_x f(x) \partial_x g(x) dx$$

one can show that \mathcal{H} is a Hilbert space of functions on Ω and that the set

$$\{\phi_n(t)\}_{n \in \mathbb{N}^*} = \left\{ \frac{\sqrt{2}}{n\pi} \sin(n\pi t) \right\}_{n \in \mathbb{N}^*}$$

is an orthonormal basis of \mathcal{H} (Debnath and Mikusinski, 1998; Atteia and Gaches, 1999). Hence, $\{\phi_n(x)\}_{n \in \mathbb{N}^*}$ is a tight frame of \mathcal{H} with the frame constant A equals to 1. Let us show that this frame verify the condition given in Theorem (5) in order to prove that \mathcal{H} is an RKHS.

At first, let us prove that for all $t \in \Omega$, the sequence $\{\phi_n(t)\}_{n \in \mathbb{N}^*}$ belongs to ℓ_2 . Because $\bar{\phi}_n = \phi_n$, we have for any $t \in \Omega$:

$$\begin{aligned} \sum_{n \in \mathbb{N}^*} \phi_n^2(t) &= \sum_{n \in \mathbb{N}^*} \bar{\phi}_n^2(t) = \sum_{n \in \mathbb{N}^*} \frac{2}{n^2 \pi^2} \sin^2(n\pi t) \\ &\leq \frac{2}{\pi^2} \sum_{n \in \mathbb{N}^*} \frac{1}{n^2} \\ &< \infty. \end{aligned}$$

Hence, according to the adjoint frame operator U^* given in equation (6), for any $t \in \Omega$, the function $\sum_{n \in \mathbb{N}^*} \phi_n(\cdot) \phi_n(t)$ is a well-defined function of \mathcal{H} . Thus,

$$\left\| \sum_{n \in \mathbb{N}^*} \phi_n(\cdot) \phi_n(t) \right\|_{\mathcal{H}}^2 = \sum_{n \in \mathbb{N}^*} \phi_n^2(t) < \infty.$$

Hence \mathcal{H} is a infinite dimensional RKHS with kernel

$$\forall s, t \in \Omega, K(s, t) = \sum_{n=1}^{\infty} \frac{2}{n^2 \pi^2} \sin(n\pi s) \sin(n\pi t).$$

Example 5 This other example shows a way for constructing an infinite dimensional RKHS from its frame. Let $\{\alpha_n\}_{n \in \Gamma}$ be a set of strictly positive real values and define the subspace ℓ_{α}^2 of ℓ^2 as

$$\ell_{\alpha}^2 = \left\{ c = \{c_n\}_{n \in \Gamma}, c_n \in \mathbb{R} : \sum_{n \in \Gamma} \frac{c_n^2}{\alpha_n} < \infty \right\}.$$

Endowed with the inner product $\langle c, d \rangle_{\ell_{\alpha}^2} \equiv \sum_{n \in \Gamma} \frac{c_n d_n}{\alpha_n}$, one can show that ℓ_{α}^2 is a Hilbert space. Now, let $\{\phi_n\}_{n \in \Gamma}$ be a set of functions on \mathbb{R}^{Ω} so that:

$$\forall t \in \Omega, \sum_{n \in \Gamma} \alpha_n \phi_n^2(t) < \infty$$

and T the mapping:

$$T : \begin{matrix} \ell_\alpha^2 & \rightarrow & \mathcal{H} \subset \mathbb{R}^\Omega \\ c & \rightarrow & f = \sum_{n \in \Gamma} c_n \phi_n. \end{matrix}$$

It is simple to show that \mathcal{H} is a space of functions on Ω since for all $t \in \Omega$, $\{\alpha_n \phi_n(t)\}_{n \in \Gamma}$ belongs to ℓ_α^2 . Then we have,

$$\langle c, \{\alpha_n \phi_n(t)\}_{n \in \Gamma} \rangle_{\ell_\alpha^2} = \sum_{n \in \Gamma} \frac{c_n \alpha_n \phi_n(t)}{\alpha_n} = \sum_{n \in \Gamma} c_n \phi_n(t) < \infty.$$

Suppose furthermore for simplicity and clarity that $\{\phi_n\}_{n \in \Gamma}$ has been chosen so that T is an injective mapping. Then the range of the mapping T also defined as

$$\mathcal{H} = \left\{ f = \sum_{n \in \Gamma} c_n \phi_n : \{c_n\}_{n \in \Gamma} \in \ell_\alpha^2 \right\}$$

and endowed with the inner product:

$$\langle f, g \rangle_{\mathcal{H}} \equiv \langle c, d \rangle_{\ell_\alpha^2} = \sum_{n \in \Gamma} \frac{c_n d_n}{\alpha_n} \quad \text{with } f = \sum_{n \in \Gamma} c_n \phi_n \text{ and } g = \sum_{n \in \Gamma} d_n \phi_n$$

is also a Hilbert space since in this case T is an isometric isomorphism between ℓ_α^2 and \mathcal{H} (Debnath and Mikusinski, 1998). Note that this way of building a Hilbert space is also described by Opfer (2004a) and Amato et al. (2004). However, none of them has presented the following frame-based point of view for showing that under some weak hypothesis \mathcal{H} can be an RKHS.

At first, note that due to the one-to-one mapping between ℓ_α^2 and \mathcal{H} , the following equality holds:

$$\forall k, n \in \Gamma, \quad \langle \phi_k, \phi_n \rangle_{\mathcal{H}} = \frac{\delta_{k,n}}{\alpha_k}$$

where $\delta_{k,n}$ is the Kronecker symbol.

Let us show that $\{\phi_n\}_{n \in \Gamma}$ is a frame of \mathcal{H} . Owing to the above property, we have $\sum_{n \in \Gamma} |\langle f, \phi_n \rangle|^2 = \sum_{n \in \Gamma} \frac{c_n^2}{\alpha_n^2}$ and $\|f\|_{\mathcal{H}}^2 = \sum_{n \in \Gamma} \frac{c_n^2}{\alpha_n}$, then it is clear that the following inequality holds:

$$\frac{1}{\alpha_{max}} \|f\|_{\mathcal{H}}^2 \leq \sum_{n \in \Gamma} |\langle f, \phi_n \rangle|^2 \leq \frac{1}{\alpha_{min}} \|f\|_{\mathcal{H}}^2$$

where $\alpha_{max} = \max_{n \in \Gamma} \alpha_n$ and $\alpha_{min} = \min_{n \in \Gamma} \alpha_n$. Since according to the frame property given in Equation (8), each frame element can be expanded as $\phi_k = \sum_{n \in \Gamma} \langle \phi_k, \phi_n \rangle \bar{\phi}_n$, we have $\phi_k = \frac{1}{\alpha_k} \bar{\phi}_k$. Hence since the frame and dual frame elements are so that for any $t \in \Omega$, we have

$$\sum_{n \in \Gamma} \alpha_n \phi_n^2(t) = \sum_{n \in \Gamma} \frac{(\bar{\phi}_n(t))^2}{\alpha_n} < \infty. \tag{24}$$

Then $\{\bar{\phi}_n(t)\}_{n \in \Gamma} \in \ell_\alpha^2$ and consequently, the function $K(\cdot, t) = \sum_{n \in \Gamma} \bar{\phi}_n(t) \phi_n(\cdot)$ is well-defined, belongs by construction to \mathcal{H} and is so that

$$\left\| \sum_{n \in \Gamma} \bar{\phi}_n(t) \phi_n(\cdot) \right\|_{\mathcal{H}} < \infty,$$

and \mathcal{H} is an RKHS whose kernel is

$$K(s, t) = \sum_{n \in \Gamma} \bar{\phi}_n(t) \phi_n(s) = \sum_{n \in \Gamma} \alpha_n \phi_n(s) \phi_n(t).$$

A practical example of such an infinite dimensional RKHS can be obtained as follows. Let us consider that $\Omega = \mathbb{R}$ and each $\phi_n(t) = \frac{1}{\sqrt{2^J}} \varphi\left(\frac{t-n}{2^J}\right)$ with $n \in \mathbb{Z}$, $J \in \mathbb{Z}$ and $\varphi(t)$ a pointwise-defined on Ω and compactly supported function so that $\{\phi_n\}_{n \in \Gamma}$ are linearly independent. Examples of such functions $\varphi(t)$ are functions that are classically used in wavelet-based multiresolution analysis (Mallat, 1998). Since each ϕ_n is a compactly supported shift of a function φ , for any t , the sum in Equation (24) becomes a finite sum of non-zero terms which convergence is consequently guaranteed for any $\{\alpha_n\}_{n \in \Gamma}$. At this point, we can state that the space

$$\mathcal{H} = \left\{ f = \sum_{n \in \Gamma} \frac{c_n}{\sqrt{2^J}} \varphi\left(\frac{t-n}{2^J}\right) : \sum_{n \in \Gamma} \frac{c_n^2}{\alpha_n} < \infty \right\}$$

is a reproducing kernel Hilbert space.

If we want \mathcal{H} to be the span of different dilations and shifts of φ , we can also show that \mathcal{H} is an RKHS by choosing the $\{\alpha_n\}_{n \in \Gamma}$ to be related to the dilation parameter J so that the inequality in (24) holds.

4.2 Other Classes of Frame-Based Kernels

Recently, Gao et al. (2001) have proposed another class of frame-based kernels. Their approach is based on the connection between regularization operator and support vector kernel as described in Smola et al. (1998). Supposing that U is the frame operator of a either finite or infinite dimensional RKHS, their kernel is based on the statement that the operator $Q = U^*U$ is a symmetric positive definite operator and the Green function associated to this operator is a Mercer kernel. Thus, the kernel they proposed, named the frame operator kernel, can be expanded with respect to the dual frame elements as

$$K(s, t) = \sum_{n \in \Gamma} \bar{\phi}_n(s) \bar{\phi}_n(t).$$

A detailed proof of this equation is given in Gao et al. (2001).

From the point of view of the regularization theory (Smola et al., 1998), this frame-operator kernel of Gao et al. is different from the one we propose as the regularization operator associated to each of them are different. In fact, in our case the regularization operator can be considered as the projector of any function space on \mathcal{H} whereas in the Gao et al. case, it can be seen as the frame operator U .

More recently, Opfer (2004b) has shown that the kernel associated to an RKHS \mathcal{H} can be expanded as

$$K(s, t) = \sum_{n \in \Gamma} \phi_n(s) \phi_n(t)$$

if and only if the set of functions $\{\phi_n\}_{n \in \Gamma}$ is a super tight frame (which is a tight frame with frame bounds equal to 1) of \mathcal{H} . This results is a particular case of Theorem (7) since for a super tight frame each dual frame element is $\bar{\phi}_n = \phi_n$. Furthermore, compared to Opfer's work, our Theorem (5) gives a frame-based condition for a Hilbert space to be an RKHS.

The works of Amato et al. (2004) and Opfer (2004a) where they both proposed the concept of multiscale kernels can also be related to our work. Interestingly, they have both shown that a Hilbert space spanned by wavelet can be under some weak hypotheses an RKHS. The way they build their RKHS \mathcal{H} is very similar to the one we described in example (5) and the related reproducing kernel is naturally

$$K(s, t) = \sum_{n \in \Gamma} \alpha_n \phi_n(s) \phi_n(t),$$

where each α_n is a strictly positive real value. On one hand, Amato et al. ended up with this kernel by considering that $\{\phi_n\}_{n \in \Gamma}$ are a orthonormal wavelet basis of $L_2([0, 1])$ and showing that for their space \mathcal{H} , the evaluation functional is continuous. On the other hand, for achieving this result, Opfer has shown that the function $K(\cdot, t)$ belongs to \mathcal{H} and satisfies the reproducing property without explicit explanations on how this kernel has been obtained. Hence, although very similar to the work of Opfer, the example (5) gives the functional setting on how the kernel in (Opfer, 2004a) can be derived.

5. Discussions

Propositions presented in previous sections describe a way for easily building RKHS and its associate reproducing kernel. Hence, this kernel can be used within the framework of regularization networks or SVMs for functional estimation.

For SVMs, one usually chooses as a kernel a continuous symmetric function K in $L_2(\Omega)$ (Ω being a compact subset of \mathbb{R}^d) that has to satisfy the following condition, known as Mercer’s condition:

$$\int_{\Omega} \int_{\Omega} K(x, y) f(x) f(y) dx dy \geq 0 \tag{25}$$

for all $f \in L_2(\Omega)$.

Now, one may ask what are the advantages and drawbacks of using kernels built by means of Theorem (5) or (8).

- Both Mercer’s condition and frameable RKHS allow to obtain a positive definite function. However, it is obvious that conditions for having frameable RKHS are easier to verify than Mercer’s condition. Thus, this can be interpreted as a flexibility for adapting kernel to a particular problem. Examples of this flexibility will be given below within the context of semiparametric estimation. Notice that methods for choosing the appropriate frame elements of the RKHS are not given here.

Example 6 Consider the set of functions on \mathbb{R} $\left\{ \phi_n(s) = \frac{\sin(\pi(s-n))}{\pi(s-n)} \right\}_{n=1 \dots N}$. The space spanned by these frame elements associated to $L_2(\mathbb{R})$ inner product is an RKHS. Thus, as a direct corollary of Theorem 8, the kernel

$$K(s, t) = \sum_{i=1}^N \bar{\phi}_i(s) \phi_i(t)$$

is an admissible kernel for SVMs.

A representation of such a kernel with $N = 9$ is given in Figure (2).

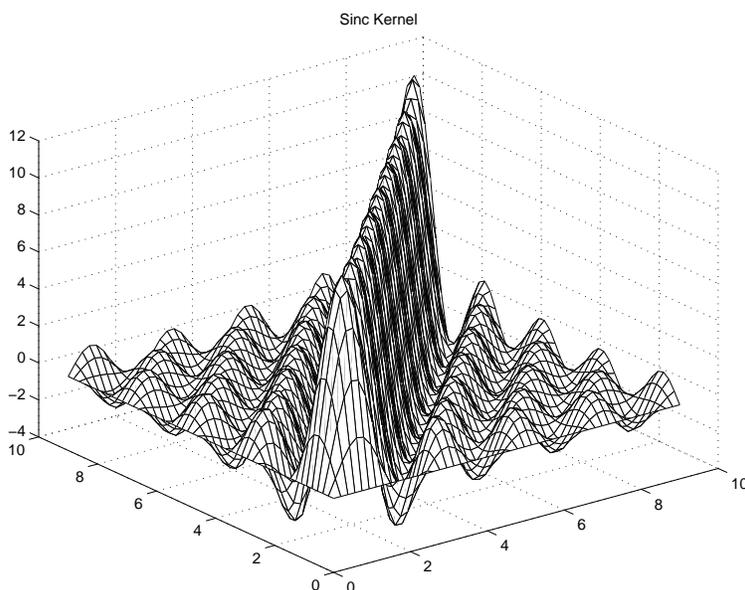


Figure 2: The sinc kernel.

- Since conditions for obtaining a frameable RKHS hold mainly for finite dimensional space (although, it may exist infinite dimensional Hilbert space which frame elements satisfy hypotheses of Theorem (5)), it is fairest to compare the frameable kernel to a finite dimensional kernel. According to Mercer's condition, or other more detailed papers on the subject (Aronszajn, 1950; Wahba, 2000), Mercer's kernel can be expanded as follows:

$$K(s, t) = \sum_{n=1}^N \frac{1}{\lambda_n} \psi_n(s) \psi_n(t)$$

where s and t belong to Ω , λ_l is a positive real number and $\{\psi_l\}_{l=1..N}$ is a set of orthogonal functions. Conditions for constructing frameable kernel are less restricting since the orthogonality of the frame elements are not needed. One can note that for tight frame or orthonormal basis, frameable kernel leads to the following expansion:

$$K(s, t) = \sum_{n=1}^N \frac{1}{A} \psi_n(s) \psi_n(t)$$

since dual frame elements is equal to frame elements up to a multiplicative constant depending on the frame bound A . Tightness of a frame is a very interesting property since in this case processing the dual frame is no more needed. However, unless we explicitly build the RKHS \mathcal{H} so that it is spanned by a tight frame (as in example (5) or in Opfer (2004b)), tightness of a frame needs more constraints on the frame elements than other frames. Thus a tight frame of a space is harder to build than other frame of the same space.

- The conditions for a frameable Hilbert space being an RKHS is given in Equation (13) and they hold also for infinite dimensional case for which the kernel is written

$$K(s, t) = \sum_n \bar{\phi}_n(s) \phi_n(t).$$

Again in this case, the frame kernel expansion is similar to the Mercer's kernel one. The main difference between the finite and infinite dimensional case relies on the fact that a finite set of functions $\{\phi_n\}$ is always a frame of the space it spans (provided that this latter is endowed with an adequate inner product). This is not always true for an infinite set of functions. However, we have shown in example (5) that under some mild conditions, it is possible to build an infinite dimensional RKHS.

- In the SVMs algorithm, the kernel realizes the dot product of the data points mapped in some feature space:

$$K(s, t) = \langle \Phi(s), \Phi(t) \rangle$$

with Φ being the mapping. Usually, this mapping is not explicitly given since one only needs for computing the optimal hyperplane the dot product in the feature space. With frame-based kernels, we have the relation

$$\begin{aligned} K(s, t) &= \sum_{n=1}^N \bar{\phi}_n(s) \phi_n(t) \\ &= \sum_{n=1}^N \bar{\phi}_n(s) \sum_{j=1}^N \bar{\phi}_j(t) \langle \phi_j(\cdot), \phi_n(\cdot) \rangle_{\mathcal{H}} \quad \text{according to Equation (10)} \\ &= \left\langle \sum_{n=1}^N \bar{\phi}_n(s) \phi_n(\cdot), \sum_{j=1}^N \bar{\phi}_j(t) \phi_j(\cdot) \right\rangle_{\mathcal{H}}. \end{aligned}$$

Thus the data embedding can be defined as

$$\Phi: \begin{array}{l} \Omega \longrightarrow \mathcal{H} \\ t \longrightarrow \sum_{n=1}^N \bar{\phi}_n(t) \phi_n(\cdot). \end{array}$$

The data points are mapped to a function belonging to \mathcal{H} . The mapping is consequently strictly related to the frame elements $\{\phi_n\}$ and is implicitly defined by them.

- Besides, since the kernel has an expansion with regards to the frame elements, the solution of Equation (1) is of easier interpretation. Indeed, although the solution depends on the kernel expression, it can be rewritten as a linear combination of the frame elements. Thus, compared to other kernels for which basis functions are unknown, using frame-based kernel increases model interpretability.
- Drawbacks of using frame-based kernel rely mainly on the time complexity burden that is added for constructing the data model. For both SVMs and regularization networks, one has

to process the kernel matrix K with elements $K_{i,j} = K(x_i, x_j)$. Thus, with frame-based kernel, one has to compute the dual frame elements, (for instance, by means of an iterative algorithm, as the one described in (Grochenig, 1993)). This by its own may be time-consuming. Furthermore, the construction of the matrix K needs the processing of the sum. Hence, if the number N of frame elements describing the kernel and the number ℓ of data are large, building K becomes rapidly very time-consuming (of an order of $N^2 \cdot \ell^2$).

Some of these points suggest that frame-based kernels can be useful by themselves. However, within the context of semiparametric estimation, this flexibility for building kernel offers some other interesting perspectives. Semiparametric estimation can be introduced by the following theorem.

Theorem 9 (Kimeldorf and Wahba, 1971)

Let \mathcal{H}_K be an RKHS of real valued functions on Ω with reproducing kernel K . Denote by $\{(x_i, y_i)_{i=1 \dots \ell}\}$ the training set and let $\{g_j, j = 1 \dots N\}$ be a set of functions on Ω such that the matrix $G_{i,j} = g_j(x_i)$ has maximal rank. Then, the solution to the problem

$$\min_{f \in \text{span}(g) + h, h \in \mathcal{H}_K} \frac{1}{\ell} \sum_{i=1}^{\ell} C(y_i, f(x_i)) + \lambda \|f\|_{\mathcal{H}_K}^2 \tag{26}$$

has a representation of the form

$$f(\cdot) = \sum_{i=1}^{\ell} c_i K(x_i, \cdot) + \sum_{j=1}^N d_j g_j(\cdot).$$

■

The solution of this problem can be interpreted as a semiparametric estimation since one part of the solution (the first sum) comes from a non-parametric estimation (the regularization problem) while the other term is due to the parametric expansion (the span of $\{g_j\}$). As stated by Smola in his thesis (Smola, 1998), semiparametric estimation can be advantageous with regards to a fully non parametric estimation as it exploits some prior knowledge on the estimation problem (for instance major properties of the data are described by linear combination of a small set of functions), and making a “good” guess (on the set of functions $\{g_j\}$) can have a large effect on performance.

Again in this context, the flexibility of frame-based kernel can be exploited. In fact, let $G = \{g_i\}_{i=1 \dots N}$ be a set of N linearly independent functions that satisfies Theorem 8, hence, any subset of G , $\{g_i\}_{i \in \Gamma}$, (Γ being an index set of size $n_0 < N$) can be used for building an RKHS \mathcal{H}_K while the remaining vectors can be used in the parametric part of the Kimeldorf-Wahba theorem. Hence in this case, the solution of (26) is written

$$f(\cdot) = \sum_{i=1}^{\ell} c_i \sum_{k \in \Gamma} \bar{g}_k(x_i) g_k(\cdot) + \sum_{j \in C_{\Gamma}} d_j g_j(\cdot).$$

The flexibility comes from the fact that in a learning problem, any elements of G can be regularized (if involved in the span of \mathcal{H}_K) or can be kept as it is (if used in the parametric part). Intuitively, one should use any vector that comes from “good” prior knowledge, in the parametric part of the approximation while leaving in the kernel expansion the other frame elements. Notice also that only

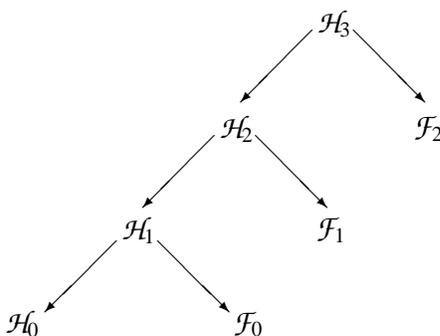


Figure 3: Example of multiscale approximation on 3 levels. Each space \mathcal{H}_j can be decomposed in a trend space \mathcal{H}_{j-1} and a detail space \mathcal{F}_{j-1} . In this case, \mathcal{H}_3 can be considered as the sum of \mathcal{H}_0 , \mathcal{F}_0 , \mathcal{F}_1 and \mathcal{F}_2 .

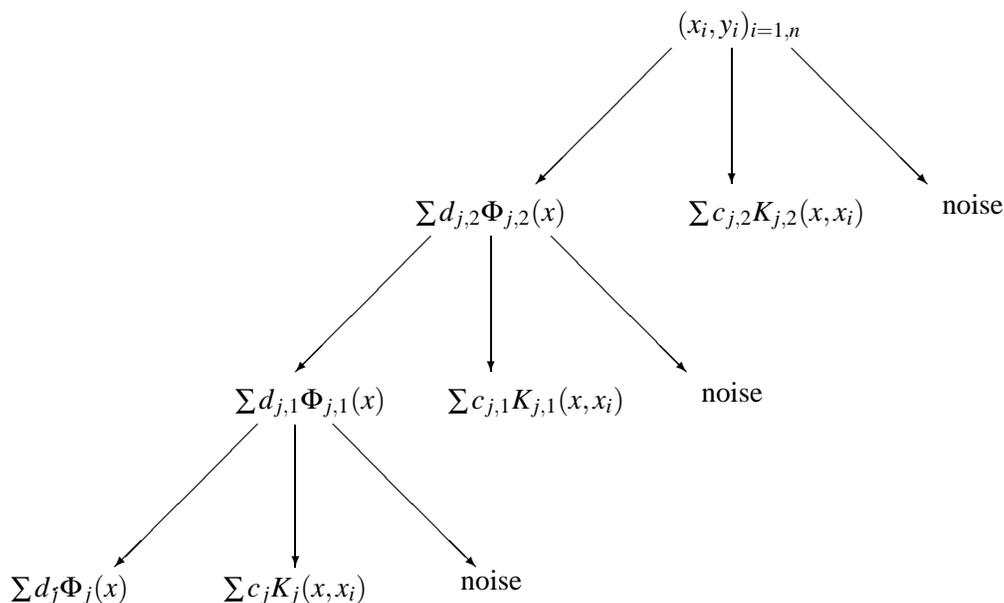


Figure 4: Example of multiscale approximation on 3 levels: the kernel point of view. For instance, here we want to learn a function $f(x)$ that has generated the samples $(x_i, y_i)_{i=1, n}$ under some noisy condition. The first step consists of decomposing the hypothesis space into a parametric part spanned by $\{\Phi_{j,2}(x)\}$ and a non parametric part spanned by $K_{j,2}(x, x_i)$. Then the resulting parametric approximation is decomposed again in two parts and so on. The multiscale approximation of $f(x)$ is then $\hat{f}(x) = \sum d_j \Phi_j(x) + \sum c_j K_j(x, x_i) + \sum c_{j,1} K_{j,1}(x, x_i) + \sum c_{j,2} K_{j,2}(x, x_i)$.

the subset of G which is used in the parametric part has to be linearly independent.

Another perspective which follows directly from this finding is a technique of regularization that we call multiscale regularization which is inspired from the multiresolution analysis of Mallat (1998). Here, we just sketch the idea behind this concept and in no way, the following paragraph should be considered a complete study of this new technique since the analysis of its properties goes beyond the scope of this paper. Consider the same problem as the one described in Theorem 9. Now, suppose that $\{g_i\}$ is a set of N linearly independent functions verifying Theorem (8). Let $\{\Gamma_i\}_{i=0\dots m}$ be a set of index set such that $\cup_{i=0}^m \Gamma_i = \{1, \dots, N\}$ and $\Gamma_i \cap \Gamma_j = \emptyset$ for $i \neq j$ and \mathcal{H} being the RKHS spanned by $\{g_i\}$. By subdividing the set $\{g_i\}$ with the index set $\{\Gamma_i\}_{i=0\dots m}$, one can construct m RKHS $\{\mathcal{F}_i\}_{i=0\dots m-1}$ in such a way that

$$\forall i = 1 \dots m, \quad \mathcal{F}_{i-1} = \text{span} \{g_k\}_{k \in \Gamma_i}$$

and reproducing kernel of \mathcal{F}_i is noted K_i . Now, denote as \mathcal{H}_i the RKHS such that

$$\forall i = 1 \dots m, \quad \mathcal{H}_i = \mathcal{H}_{i-1} + \mathcal{F}_{i-1}$$

with $\mathcal{H}_0 = \text{span} \{g_k\}_{k \in \Gamma_0}$. By construction, the space \mathcal{H}_i are nested spaces:

$$\mathcal{H}_0 \subset \mathcal{H}_1 \subset \dots \subset \mathcal{H}_m = \mathcal{H}.$$

In this case, one can interpret \mathcal{H}_0 as the space of lower approximation capacity whereas \mathcal{H}_m is the space with higher capacity. Besides, since $\mathcal{H}_i = \mathcal{H}_{i-1} + \mathcal{F}_{i-1}$, one can think of \mathcal{F}_{i-1} as the details needed to be added to \mathcal{H}_{i-1} to obtain \mathcal{H}_i , thus we will call spaces \mathcal{F}_i the “details” spaces whereas spaces \mathcal{H}_i are the “trend” spaces. Every of these spaces \mathcal{F}_i and \mathcal{H}_i are an RKHS since any subset of $\{g_i\}$ satisfies Theorem (8).

Multiscale regularization is an iterative technique that at step $k = 1, \dots, m$ consists of looking for the solution $f_{m-k}(\cdot)$ of the following minimization problem:

$$\min_{f \in \mathcal{H}_{m-k+1}} \frac{1}{n} \sum_{i=1}^n C(y_{i,m-k}, f(x_i)) + \lambda_{m-k} \|f\|_{\mathcal{H}_{m-k}}^2 \tag{27}$$

where $y_{i,m-1} = y_i$, $y_{i,m-(k+1)} = y_{i,m-k} - \sum_{j=1}^n c_{j,m-k} K_{m-k}(x_j, x_i)$. According to the representer Theorem (9), $f_{m-k}(\cdot)$ can be written:

$$f_{m-k}(\cdot) = \sum_{i=1}^n c_{i,m-k} K_{m-k}(x_i, \cdot) + \sum_{j \in \cup_{i=0}^{m-k} \Gamma_i} d_{j,m-k} g_j(\cdot) \tag{28}$$

and thus the overall solution of the so-called multiscale regularization is

$$\hat{f}(\cdot) = \sum_{k=1}^m \sum_{i=1}^n c_{i,m-k} K_{m-k}(x_i, \cdot) + \sum_{j \in \Gamma_0} d_{j,0} g_j(\cdot). \tag{29}$$

The solution \hat{f} of the multiscale regularization is the sum of different approximations on nested spaces. At first, one seeks to approximate the data on the highest approximation capacity space by regularizing only the details. Then, these details are subtracted to the data and one tries to approximate this residual on the next space by keeping regularizing the details on this space, and

so on. Thus at each step, one can control the “amount” of regularization brought to each details space, increasing in this way the capacity control capability of the model. Figure (3) and (4) show an example of how the algorithm works for a 3-level approximation scheme.

The framework of additive models of Hastie et al. (Hastie and Tibshirani (1990)) can give other insights to multiscale regularization. In fact, if we suppose that the family $\{g_i\}_{i=1,\dots,N}$ forms an orthonormal basis of \mathcal{H} and build the spaces \mathcal{H}_0 and \mathcal{F}_m in the same way as described above, then by construction, we have

$$\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{F}_0 \oplus \dots \oplus \mathcal{F}_{m-1}.$$

Hence any function $f \in \mathcal{H}$ can be written as $f(x) = \sum_{i=0}^m f_i(x)$ with $f_0 \in \mathcal{H}_0$ and $f_i \in \mathcal{F}_{i-1}$ for $i = 1, \dots, m$. Thus, the multiscale regularization algorithm can be interpreted as an algorithm which looks for the function f that minimizes the following empirical risk:

$$R_{reg}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} C(y_i, \sum_{j=0}^m f_j(x_i)) + \sum_{j=1}^m \lambda_j \|f_j\|_{\mathcal{F}_{j-1}}^2 \tag{30}$$

where each λ_j is a hyperparameter that controls the amount of regularization for \mathcal{F}_{j-1} . This minimization problem is typically the problem of fitting an additive model as proposed by Hastie and Tibshirani (1990).

Illustrations of the multiscale regularization algorithm on both toy and real-world problems are given in the next section.

6. Numerical Experiments

This section describes some experiments that compare frame-based kernels to classical one (for instance gaussian kernel) on some regression problems. Besides, illustrations of some points raised in the discussion such as the multiscale approximation algorithm are given.

6.1 Experiment 1

This first experiment aims at comparing the behavior of different kernels using regularization networks and support vector regression. The function to be approximated is

$$f(x) = \sin x + \text{sinc}(\pi(x - 5)) + \text{sinc}(5\pi(x - 2)) \tag{31}$$

where $\text{sinc}(x) = \frac{\sin x}{x}$. Data used for the approximation is corrupted by an additive noise, thus $y_i = f(x_i) + \varepsilon_i$ where ε_i is a zero-mean gaussian noise of standard deviation 0.2 . Points x_i are drawn from uniform random sampling of interval $[0, 10]$. Three kernels have been used for the approximation:

- Gaussian kernel:

$$K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

- Wavelet kernel:

$$K(x, y) = \sum_{i \in \Gamma} \tilde{\psi}_i(x) \psi_i(y)$$

where i denote a multi index and $\psi_i(x) = \psi_{j,k}(x) = \frac{1}{\sqrt{a^j}} \psi\left(\frac{x - ku_0 a^j}{a^j}\right)$. $\psi(x)$ is the mother wavelet which in this experiment is a mexican hat wavelet. Dilation parameter j takes value

	Regularization Networks	Support vector regression
Gaussian kernel	0.0218 ± 0.0049	0.0248 ± 0.0058
Wavelet kernel	0.0249 ± 0.0078	0.0291 ± 0.0086
Sin/Sinc kernel	0.0249 ± 0.0122	0.0302 ± 0.0176

Table 1: True generalization error for Gaussian, Wavelet, Sin/Sinc kernels with Regularization Networks and support vector regression for the best hyperparameters.

in the set $\{-5, 0, 5\}$ whereas k is chosen so that a given wavelet $\psi_{j,k}(x)$ has its support in the interval $[0, 10]$. For now on, we set $u_0 = 1$ and $a = 2^{0.25}$. These values are those proposed by Daubechies (Daubechies, 1992) so that a wavelet set is a frame of $L_2(\mathbb{R})$. Notice that in our case, we only use a subset of this frame.

- Sin/Sinc kernel:

$$K(x, y) = \sum_{i \in \Gamma} \bar{\phi}_i(x) \phi_i(y)$$

where $\phi_i(x) = \{1, \sin(x), \cos(x), \text{sinc}(j\pi(x-k)) : j \in \{1, 3, 6\}, k \in [1 \dots 9])\}$.

For frame-based kernel, if necessary the dual frame is processed using Grochenig’s algorithm.

For both regularization network and support vector regression, some hyperparameters have to be tuned. Different approaches are possible for solving this model selection problem. In this study, the true generalization error has been evaluated for a range of finely sampled values of hyperparameters. This is repeated for a hundred different data sets, and the mean and standard deviation of the generalization error are thus obtained. Table 1 depicts the true generalization error evaluated on 200 datapoints for the two learning machines and the different kernels using the best hyperparameters setting. Analysis of this table leads to the following observation: The different kernels and learning machines give comparable results (all averages are within one standard deviation from each other). Using prior knowledge on the problem in this context does not improve performance (Sin/Sinc kernel or wavelet kernel compared to gaussian kernel). A justification can be that such kernels use strong prior knowledge (the *sin* frame element) that is included in the kernel expansion and thus this prior knowledge gets regularized as much as other frame elements. This suggests that semiparametric regularization should be more appropriate to get advantage of such a kernel.

6.2 Experiment 2

In this experiment, we suppose that some additional knowledge on the approximation problem is available, and thus its exploitation using semiparametric approximation should lead to better performance. We have kept the same experimental setup as the one used in the first example but we have restricted our study to regularization networks.

Basis functions and kernel used are the following:

- Gaussian kernel and sinusoidal basis functions $\{1, \sin(x), \cos(x)\}$.
- Gaussian kernel and wavelet basis functions $\left\{ \psi_{j,k}(x) = \frac{1}{\sqrt{a^j}} \Psi\left(\frac{x - ku_0 a^j}{a^j}\right), j \in \{0, 5\} \right\}$

- Wavelet kernel and wavelet basis functions: these functions are the same as in the previous case but the kernel is built only with low dilation wavelet ($j = -10$). In a nutshell, we can consider that the RKHS associated to the kernel used in the non- parametric context (experiment 1) has been splitted in two RKHS. One that leads to a hypothesis space that have to be regularized and another one that does not have to be controlled.
- Sinc kernel and Sin/Sinc basis functions: in this setting, the kernel is given by the following equation:

$$K(x, y) = \sum_{i \in \Gamma} \bar{\phi}_i(x) \phi_i(y)$$

with $\phi_i(x) = \{\text{sinc}(j\pi(x - k)) : j \in \{3, 6\}, k \in [1 \dots 9]\}$

and the basis functions are $\{1, \sin x, \cos x, \text{sinc}(\pi(x - k)) : k \in [1 \dots 9]\}$.

For each kernel, model selection has been solved by cross-validation using 50 data sets. Then, after having spotted the best hyperparameters, the experiment was run a hundred times and the true generalization error in a mean-square sense, was evaluated. Table 2 summarizes all these trials and describes the performance improvement achieved by different kernels compared to the gaussian kernel and sin basis functions. From this table, one can note that:

- exploiting prior knowledge on the function to be approximated leads immediately to a lower generalization error (compare Table 1 and Table 2).
- as one may have expected, using strong prior knowledge on the hypothesis space and the related kernel gives considerably higher performances than gaussian kernel. In fact, the sinc-based kernel achieves by far the lower mean square error. The idea of including the “good” knowledge in a non-regularized hypothesis space while including the “bad” prior knowledge in the RKHS span seems to be fruitful in this case (the frame elements $\text{sinc}(3\pi(x - k))$ and $\text{sinc}(6\pi(x - k))$ can be termed as “bad” knowledge as, they are not used in the target function).
- wavelet kernel achieves minor improvement of performance compared to gaussian kernel. However, this is still of interest as using wavelet kernel and basis functions does corresponds to prior knowledge that can be reformulated as: “the function to be approximated contains smooth structure (the *sin* part), irregular structures (the *sinc* part) and noise”. It is obvious that knowing the true basis function leads to better performance, however that information is not always available and using bad knowledge may result in poorer performance. Thus, prior knowledge on structures which may be easiest to get than prior knowledge on basis function can be easily exploited by means of wavelet span and wavelet kernel.

6.3 Experiment 3

This last simulated example targets at illustrating the concept of multiscale regularization. We have compared several learning algorithms in function approximation problems. The learning machines are: regularization networks, SVM, semiparametric regularization and multiscale regularization. For the two first methods, a gaussian kernel is used whereas for the two latter, wavelet kernel

Kernel / Basis Functions	M.S.E	Improvement (%)
Gaussian / Sin	0.0216 ± 0.0083 (6)	0
Gaussian / Wavelet	0.0202 ± 0.0072 (4)	4.6
Wavelet / Wavelet	0.0195 ± 0.0077 (2)	9.7
Sinc / Sin	0.0156 ± 0.0076 (88)	27.8

Table 2: True generalization performance for semiparametric regression networks and different settings of kernel and basis functions. The number in parentheses reflects the number of trials for which the model has been the best model.

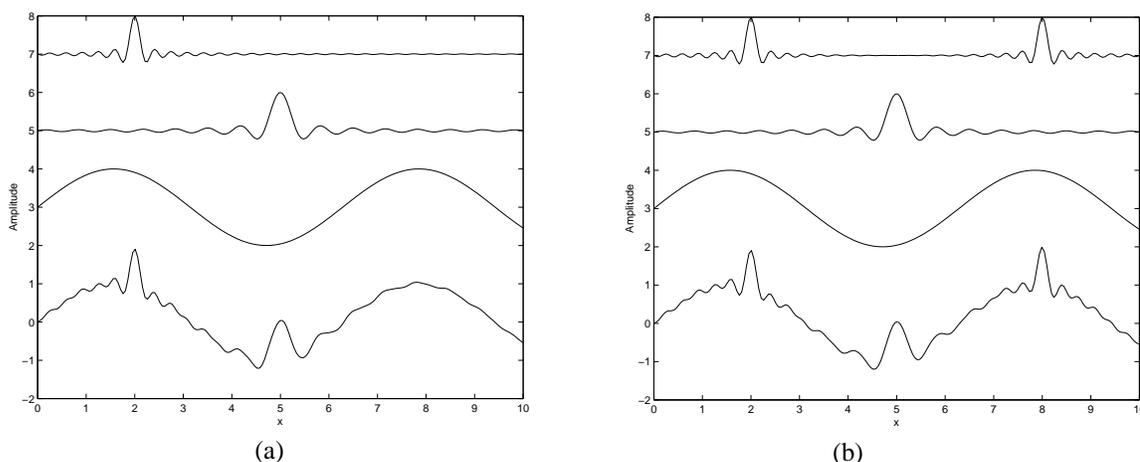


Figure 5: Original functions used for benchmarking in experiment 3. (a) f_1 (b) f_2 . Top: multiscale structure on 3 levels. Bottom: Complete function.

and basis functions are taken. The true functions used for benchmarking are the following:

$$\begin{aligned}
 f_1(x) &= \sin x + \operatorname{sinc}(3\pi(x-5)) + \operatorname{sinc}(6\pi(x-2)), \\
 f_2(x) &= \sin x + \operatorname{sinc}(3\pi(x-5)) + \operatorname{sinc}(6\pi(x-2)) + \operatorname{sinc}(6\pi(x-8)).
 \end{aligned}$$

The two functions f_1 and f_2 have been randomly sampled on the interval $[0, 10]$. Gaussian noise ε_i of standard deviation 0.2 is added to the samples, thus the entries of the learning machines become $\{x_i, f(x_i) + \varepsilon_i\}$. Here again, a range of finely sampled values of hyperparameters has been tested for model selection. In each case, an averaging of the true error generalization over 100 data sets of 200 samples was evaluated using a uniform measure.

For semiparametric regularization, the kernel and basis setting was built with a wavelet set given by

$$\Psi_{j,k}(x) = \frac{1}{\sqrt{a^j}} \Psi\left(\frac{x - ku_0 a^j}{a^j}\right).$$

	f_1	f_2
Gaussian Reg. Networks	0.0266 ± 0.0085	0.0385 ± 0.0141
Gaussian SVM	0.0328 ± 0.0093	0.0475 ± 0.0155
Semip Reg. Networks 1	0.0266 ± 0.0085	0.0397 ± 0.0113
Semip Reg. Networks 2	0.0236 ± 0.0063	0.0353 ± 0.0080
Multi. Regularization	0.0246 ± 0.0060	0.0344 ± 0.0069

Table 3: True mean-square-error generalization for regularization networks, SVM, semiparametric regularization networks, and multiscale regularization for f_1 and f_2 .

The kernel is constructed from a set of wavelet frame of dilation j_{SPH} and the basis functions are given by another wavelet set described by j_{SPL} . For multiscale regularization, the setting of the nested spaces are the following:

$$\begin{aligned}\mathcal{H}_0 &= \text{span} \left\{ \frac{1}{\sqrt{a^j}} \Psi \left(\frac{t - ku_0 a^j}{a^j} \right), j = 5 \right\}, \\ \mathcal{F}_0 &= \text{span} \left\{ \frac{1}{\sqrt{a^j}} \Psi \left(\frac{t - ku_0 a^j}{a^j} \right), j = 0 \right\}, \\ \mathcal{F}_1 &= \text{span} \left\{ \frac{1}{\sqrt{a^j}} \Psi \left(\frac{t - ku_0 a^j}{a^j} \right), j = -10 \right\}.\end{aligned}$$

These dilation parameters have been set in a *ad hoc* way, but their choices can be justified by the following reasoning: Three distinct levels have been used for separating the approximation in three structures which should be smooth ($j = 5$), irregular ($j = 0$) and highly irregular ($j = -10$). The same values of j were used in the semiparametric context. Two semiparametric settings have been tested: the first one uses $j_{SPH} = -10$ and $j_{SPL} = \{0, 5\}$ and the other one is configured as follows $j_{SPH} = \{-10, 0\}$ and $j_{SPL} = 5$.

Table 3 presents the average of the mean-square error of the different learning machines for the two functions and for the best hyperparameter value found by cross-validation. Comments and analysis of this experiment validating the concept of multiscale approximation are:

- semiparametric 2 and multiscale approximation give the best mean-square error. They achieve respectively a performance improvement with regards to gaussian regularization networks of 11.2% and 7.5% for f_1 , and 8.3% and 10.6% for f_2 . Also note that both learning machines give the lowest standard deviation of the mean square error.
- multiscale approximation balances loss of approximation due to error at each level (see Figure) and flexibility of regularization, thus its performance is better than semiparametric one's when the multiscale structure of the signal is more pronounced.
- comparison of the two semiparametric settings shows that the second setup outperforms the first one (especially for f_2). This highlights the importance of selecting the hypothesis space

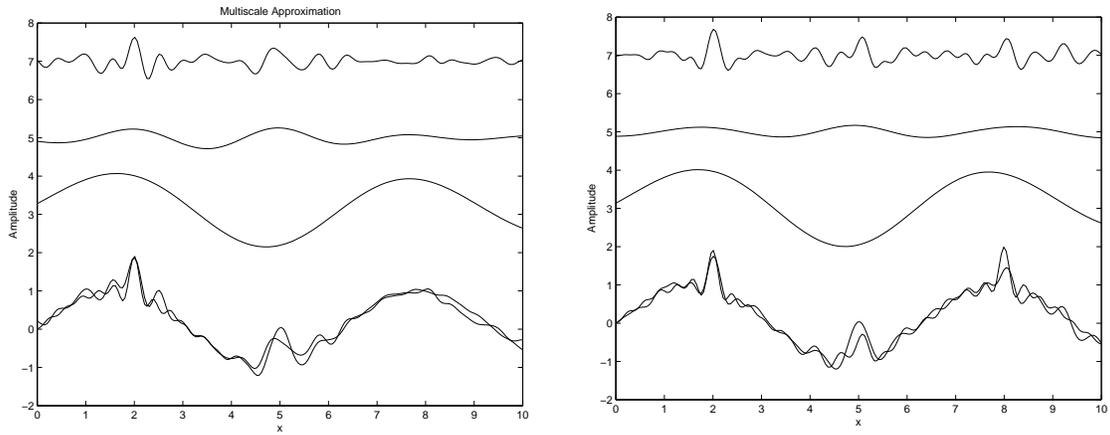


Figure 6: Top: Multiscale structure of a typical prediction of f_1 (left) and f_2 (right) by multiscale wavelet approximation Bottom: full approximation and true function

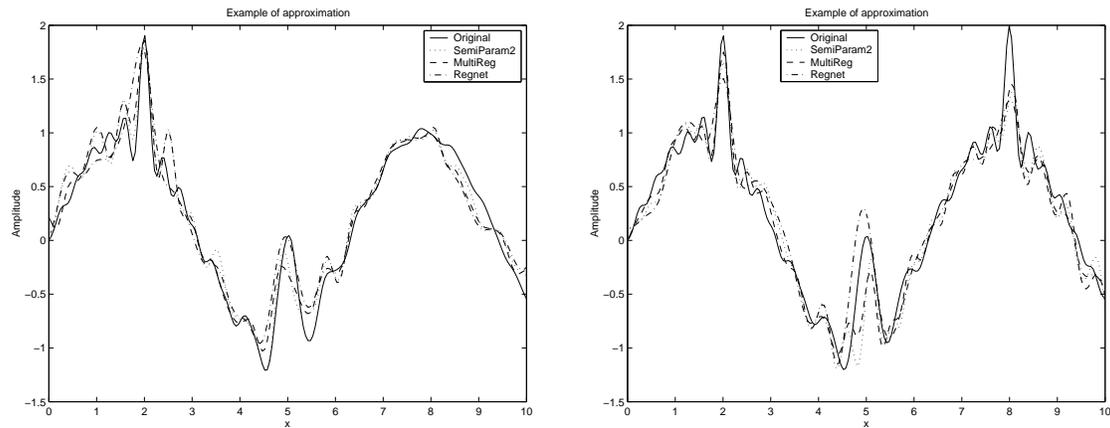


Figure 7: Examples of approximation of f_1 (left) and f_2 (right): Original function (Solid), Semi-parametric 2 (dotted), Multiscale regularization (dashed), Regularization network (dash-dotted)

to be regularized. In this experiment, it seems that leaving the space spanned by wavelet of dilation $j = 0$ on the parametric span (the space which is not regularized) leads to overfitting.

- multiscale approximation is able to catch all the structures of the signal (see Figure (7)). One can see that each level of approximation represents one structure of the function f_1 and f_2 : the lowest dilation ($j = -10$) represents the wiggles due to the highest frequency sinc, at level $j = 0$, one has the $\text{sinc}(3x)$ function whereas the sin is located on the highest dilation $j = 5$.
- Figure (6.3) shows that multiscale and semiparametric algorithms achieve better approximation of the “wiggles” than nonparametric methods without compromising smoothness in region of the functions where it is needed.

6.4 Experiments on Real-World Data Sets

This paragraph presents some estimation results on real-world time-series. These times-series are publicly available in a time-series data library (Hyndman and Akram (1998)) and have already been widely used in the field of statistics. The first one *engines* concerns a monthly measured ratio between the motor vehicles engines production and the consumer price index in Canada whereas the second one *basiron* deals with the monthly production of iron in Australia. The problem we want to solve is the estimation of these time-series after a zero-mean normalization.

For this purpose, two models have been compared, the first one being a regularization networks with a gaussian kernel whereas the other one is a multiscale regularization algorithm with an orthogonal wavelet kernel. The wavelet that has been used is a *Symmlet* wavelet with 4 vanishing moments (Mallat, 1998). The kernel of the corresponding hypothesis space \mathcal{H} which have been split into three orthogonal spaces, is so that

$$\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{F}_0 \oplus \mathcal{F}_1 \quad \text{and} \quad K_{\mathcal{H}}(x, y) = K_{\mathcal{H}_0}(x, y) + K_{\mathcal{F}_0}(x, y) + K_{\mathcal{F}_1}(x, y)$$

with

$$K_{\mathcal{H}_0}(x, y) = \sum_{j=j_{\min}}^{j_1} \sum_k \psi_{j,k}(x) \psi_{j,k}(y) + \sum_k \phi_{j,k}(x) \phi_{j,k}(y), \quad (32)$$

$$K_{\mathcal{F}_0}(x, y) = \sum_{j=j_1+1}^{j_2} \sum_k \psi_{j,k}(x) \psi_{j,k}(y), \quad (33)$$

$$K_{\mathcal{F}_1}(x, y) = \sum_{j=j_2+1}^{j_{\max}} \sum_k \psi_{j,k}(x) \psi_{j,k}(y), \quad (34)$$

and the dilation indexes are so that $j_{\min} \leq j_1 \leq j_2 \leq j_{\max}$. For both data sets, we have set $j_{\min} = -3$, $j_1 = 0$, $j_2 = 4$ and $j_{\max} = 7$.

For each estimation trial, each data set has been randomly split in a learning set of 100 samples with the remaining samples being considered as the test set. The results that we present are the normalized mean-squared error averaged over 30 trials for the best hyperparameters values of each model: for the gaussian regularization networks and the multiscale regularization networks, these hyperparameters are respectively $\{\lambda, \sigma\}$ and $\{\lambda_0, \lambda_1, \lambda_2\}$ which are the regularization parameters associated to each scale. The best hyperparameters have been obtained by evaluating the test error on a large range of finely sampled values of these hyperparameters.

	<i>basiron</i>	<i>engine</i>
Gaussian Reg. Networks	10.55 ± 1.24 (1)	37.57 ± 5.62 (8)
Multi. Regularization	9.58 ± 1.21 (29)	36.00 ± 4.30 (22)

Table 4: Averaged normalized mean-square error of estimation of real-world time-series with a gaussian and a wavelet multiscale regularization networks. The number within parenthesis is the number of time a given model has performed better than the other.

Table (4) summarizes the performance of each model. It shows that for both time-series, the multiscale algorithm performs better than the gaussian regularization networks. Indeed, for the *basiron* data set, although the difference in normalized mean-squared error is only about 0.9%, the multiscale approach has given the best results on 29 of the 30 trials. For the *engines* time-series, although the difference in normalized mean-squared error is higher (1%), our algorithm gives better results on only 22 trials. Figure (8) depicts some examples of estimation for both time-series and algorithms. This figure shows that the best model for the gaussian regularization networks is rather a smooth model whereas the wavelet multiscale model is far less smooth. This is essentially due to the nature of the time-series which are composed of a slow-varying part denoting the trend of the series, and a fast-varying part denoting the fluctuation of the time-series around the trend. Hence, because of the particular structure of the signal to be estimated, the gaussian model is not able to estimate correctly both the trend and the fluctuation whereas the multiscale model gives a better estimate. This is particularly clear for the *basiron* data set which is composed of a slow-varying trend and fluctuations.

7. Conclusions

In this paper, we showed that an RKHS can be defined by its frame elements and conversely, one can construct an RKHS from a frame. One of the key result is that the space spanned by any finite number of functions belonging to a given Hilbert space, endowed with an adequate inner product, is an RKHS with a kernel that can be at least numerically described. We have also proposed some conditions for a infinite dimensional Hilbert space to be an RKHS. These conditions depend on the frame and the dual frame elements of the Hilbert space and under some weak hypothesis, these conditions are easy to check (see example 5) . Hence, we have essentially provided some methods for building a specific kernel adapted to a problem at hand.

By exploiting this new way for constructing RKHS, a multiscale algorithm using nested RKHS has been introduced and examples given in this paper showed that using this algorithm or a semi-parametric approach with frame-based kernel improves the result of a regression problem with regards to nonparametric approximation. It has also been shown that these frame-based kernels allow better approximation only if exploited in a semiparametric context. Using them as a regularization network or SVMs kernels are not as efficient as one may have expected. However, depending on the prior knowledge on the problem, one can build appropriate kernels that can enhance the quality of the regressor within a semiparametric approach. However, for fully taking advantage of the main theorem proposed in this paper, one has to answer some open questions:

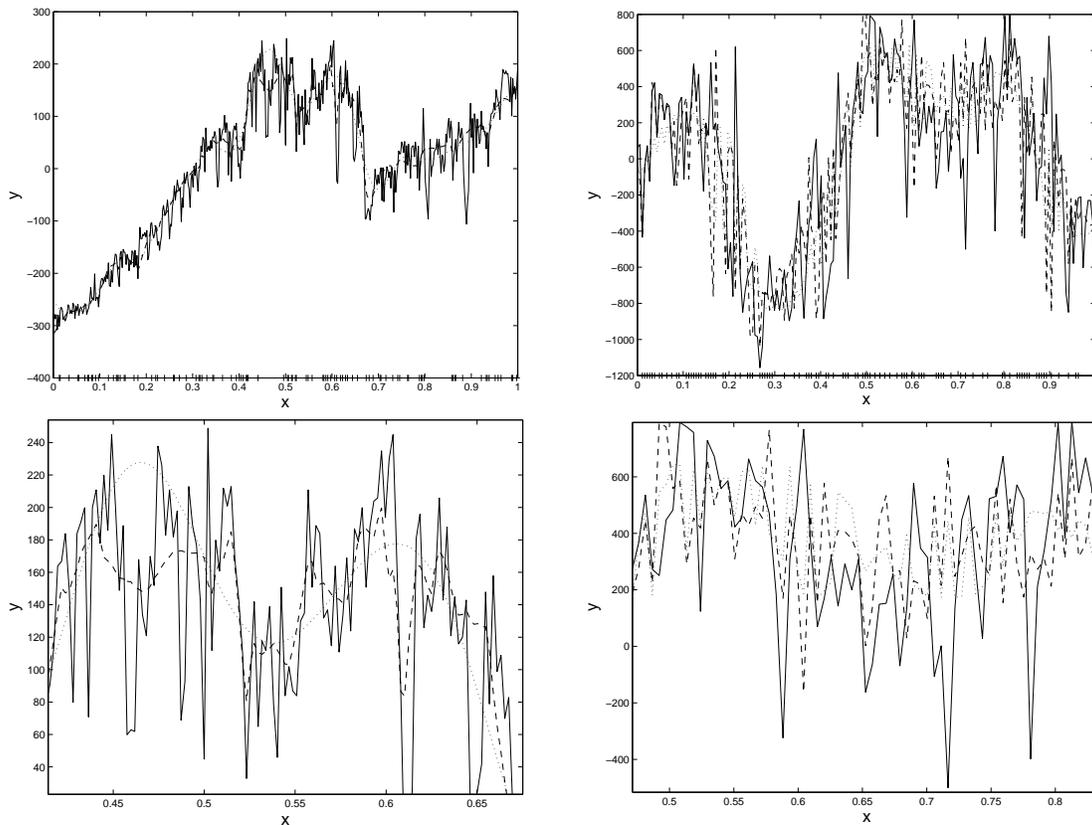


Figure 8: Examples of estimation of real-world time-series. The left and right columns respectively depict estimation of *basiron* and *engines*. The top and bottom figures respectively show the full time-series estimations and a zoomed version of these estimations. The + marks at the bottom of each figure denote the position of the learning examples in the time-series. (solid) true function. (dotted) gaussian regularization networks estimation. (dashed) wavelet multiscale regularization networks estimation.

- we give conditions for building RKHS to be used for approximation. But the difficulty stands in one question: How to transform prior information on the learning problem to frame elements? This is still an open issue.
- reconstruction from frame elements has been shown to be more robust in presence of noise (Daubechies, 1992). In fact, redundancy attenuates noise effects on the frame coefficients. Thus, this is a good statistical argument for using frame with high redundancy. However, this implies the computing of the dual frame and consequently a higher time complexity of the algorithm. Hence, fast algorithms still have to be derived.
- a multiscale regularization algorithm has been sketched in this paper in order to take advantage of frame kernels. Although some experiments show that in some situations, this algorithm performs well, it is not clear whether it theoretically sounds or not. Hence, some

further works have to be carried for a better theoretical understanding of this novel regularization method and for a better implementation of the algorithm and all the subsequent problems such as model selection.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments. This work was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

Appendix A.

We recall in this appendix a numerical method to process the dual frame of a frameable Hilbert space \mathcal{H} with frame elements $\{\phi_n\}_{n \in \Gamma}$. Let us define the operator S

$$S : \begin{cases} \mathcal{H} & \longrightarrow \mathcal{H} \\ f & \longrightarrow \sum_{n \in \Gamma} \langle f, \phi_n \rangle \phi_n. \end{cases} \quad (35)$$

One can also write the operator S as $S \triangleq U^*U$ where U is the frame operator defined in equation (5) and (6). Our goal is to process

$$\forall n, \quad \bar{\phi}_n = S^{-1}\phi_n.$$

Grochenig (1993) has proposed an algorithm to compute the problem $f = S^{-1}g$. The idea is to calculate f with a gradient descent algorithm along orthogonal directions with respect to norm induced by the symmetric operator S :

$$\|f\|_S^2 = \|Sf\|^2.$$

This norm is useful to compute the error.

Theorem 10 *Let $g \in \mathcal{H}$. To compute $f = S^{-1}g$, one has to initialize*

$$f_0 = 0, \quad r_0 = p_0 = g, \quad p_{-1} = 0.$$

Then, for any $n \geq 0$, one defines by induction,

$$\lambda_n = \frac{\langle r_n, p_n \rangle}{\langle p_n, Sp_n \rangle} \quad (36)$$

$$f_{n+1} = f_n + \lambda_n p_n \quad (37)$$

$$r_{n+1} = r_n - \lambda_n Sp_n \quad (38)$$

$$p_{n+1} = Sp_n - \frac{\langle Sp_n, Sp_n \rangle}{\langle p_n, Sp_n \rangle} p_n - \frac{\langle Sp_n, Sp_{n-1} \rangle}{\langle p_{n-1}, Sp_{n-1} \rangle} p_{n-1}. \quad (39)$$

If $\sigma = \frac{\sqrt{B}-\sqrt{A}}{\sqrt{B}+\sqrt{A}}$, then

$$\|f - f_n\|_S \leq \frac{2\sigma^n}{1 + 2\sigma^n} \|f\|_S \quad (40)$$

and thus, $\lim_{n \rightarrow +\infty} f_n = f$.

Then, in order to process numerically the dual frame of \mathcal{H} , one has to apply this algorithm on each element of the frame.

One can note that the speed of convergence is highly dependent on frame bounds A and B .

References

- U. Amato, A. Antoniadis, and M. Pensky. Wavelet kernel penalized estimation for non-equispaced design regression. *Statistics and Computing*, to appear, 2004.
- N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, (68):337–404, 1950.
- M. Atteia. *Hilbertian kernels and spline functions*. North-Holland, 1992.
- M. Atteia and J. Gaches. *Approximation hilbertienne : Splines, Ondelettes, Fractales*. Presses Universitaires de Grenoble, 1999.
- A. Berlinet and C. Thomas Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, 2004.
- B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.
- H. Brezis. *Analyse fonctionnelle, Théorie et applications*. Masson, 1983.
- C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 1998.
- O. Christensen. *Frame decomposition in Hilbert Spaces*. PhD thesis, Aarhus Univ. Denmark and Univ. of Vienna, Austria, 1993.
- I. Daubechies. *Ten Lectures on Wavelet*. SIAM, CBMS-NSF regional conferences edition, 1992.
- L. Debnath and P. Mikusinski. *Introduction to Hilbert Spaces with applications*. Academic Press, 1998.
- R. Duffin and A. Schaeffer. A class of nonharmonic fourier series. *Trans. Amer. Math. Soc.*, 72: 341–366, 1952.
- T. Evgeniou, M. Pontil, and T. Poggio. Regularization Networks and Support Vector Machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.
- J. Gao, C. Harris, and S. Gunn. On a class of a support vector kernels based on frames in function hilbert spaces. *Neural Computation*, 13(9):1975–1994, 2001.
- F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.
- F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.
- K. Grochenig. Acceleration of the frame algorithm. *IEEE Trans. Signal Proc.*, 41(12):3331–3340, 1993.
- C. Groetsch. *Inverse Problems in the mathematical sciences*. Vieweg and Sohn, 1993.

- T. Hastie and R. Tibshirani. *Generalized additive models*. Chapman and Hall, 1990.
- R. Hyndman and M. Akram. Time Series data Library. Technical report, University of Monash, Dept of Econometrics and Business Statistics, 1998. <http://www-personal.buseco.monash.edu.au/hyndman/TSDL/index.htm>.
- T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*, 1999.
- G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.
- S. Mallat. *A wavelet tour of signal processing*. Academic Press, 1998.
- V. Morosov. *Methods for solving incorrectly posed problems*. Springer Verlag, 1984.
- P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior information in machine learning by creating virtual examples. In *Proceedings of the IEEE*, volume 86, pages 2196–2209, 1998.
- R. Opfer. Multiscale kernels. Technical report, Institut für Numerische und Angewandte Mathematik, Universität Göttingen, 2004a.
- R. Opfer. Tight frame expansions of multiscale reproducing kernels in Sobolev spaces. Technical report, Institut für Numerische und Angewandte Mathematik, Universität Göttingen, 2004b.
- B. Scholkopf, P. Y. Simard, A. J. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural information processing systems*, volume 10, pages 640–646, Cambridge, MA, 1998. MIT Press.
- A. Smola. *Learning with Kernels*. PhD thesis, Published by: GMD, Birlinghoven, 1998.
- A. Smola, B. Scholkopf, and KR Muller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.
- A. Tikhonov and V. Arsénin. *Solutions of Ill-posed problems*. W.H. Winston, 1977.
- V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y, 1995.
- V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- V. Vapnik, S. Golowich, and A. Smola. *Support Vector Method for function estimation, Regression estimation and Signal processing*, volume Vol. 9. MIT Press, Cambridge, MA, neural information processing systems, edition, 1997.
- G. Wahba. *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, 1990.
- G. Wahba. An introduction to model building with reproducing kernel hilbert spaces. Technical Report TR-1020, University of Wisconsin-Madison, 2000.

Local Propagation in Conditional Gaussian Bayesian Networks

Robert G. Cowell

RGC@CITY.AC.UK

Faculty of Actuarial Science and Statistics

Sir John Cass Business School

106 Bunhill Row

London EC1Y 8TZ, U.K.

Editor: Craig Boutilier

Abstract

This paper describes a scheme for local computation in conditional Gaussian Bayesian networks that combines the approach of Lauritzen and Jensen (2001) with some elements of Shachter and Kenley (1989). Message passing takes place on an elimination tree structure rather than the more compact (and usual) junction tree of cliques. This yields a local computation scheme in which all calculations involving the continuous variables are performed by manipulating univariate regressions, and hence matrix operations are avoided.

Keywords: Bayesian networks, conditional Gaussian distributions, propagation algorithm, elimination tree

1. Introduction

Bayesian networks were developed within the field of artificial intelligence as a tool for representing and managing uncertainty (Pearl, 1988; Cowell et al., 1999), but are now finding many applications beyond that field. When a Bayesian network represents the joint distribution of a set of finite discrete random variables, exact and efficient local computations schemes may be used to evaluate marginal distributions of interest. Shachter and Kenley (1989) introduced Gaussian influence diagrams to represent multivariate Gaussian distributions and performed inference on them using standard influence diagram operations such as arc-reversals and barren node removal. Raphael (2003) presented an alternative computational scheme for degenerate multivariate Gaussian distributions and has applied it to problems of rhythmic parsing of music.

Lauritzen (1992) introduced a method of exact local computation of means and variances for Bayesian networks with conditional Gaussian distributions (Lauritzen and Wermuth, 1984, 1989), but it was later discovered that the method was numerically unstable. More recently Lauritzen and Jensen (2001) have developed an alternative and stable local computation scheme in junction trees for these conditional Gaussian networks. Apart from the improved numerical stability compared to the previous algorithm, their method is able to calculate full mixture marginals of continuous variables, and is also able to include deterministic linear relationships between continuous variables. However their method is quite complicated, requiring evaluations of matrix generalized inverses, and recursive combinations of potentials. This paper presents an alternative scheme in which the local computation is performed on an elimination tree, rather than using a junction tree. As will be shown this means that matrix manipulations are avoided because all message passing opera-

tions involving the densities of the continuous variables are performed by manipulating univariate regressions, and complex operations such as recursive combination of potentials are avoided.

The plan of the paper is as follows. The following section presents notation for conditional Gaussian regressions. Then elimination trees are defined and compared to junction trees. A simple network is used to illustrate the various phases of the message passing scheme. Then the general procedure is presented: deriving an elimination tree derived from a network; initializing the elimination tree is given (this is perhaps the most complicated part of the scheme); entering evidence and evaluating posterior marginal densities. A discussion relating the current scheme to those of Shachter and Kenley (1989) and Lauritzen and Jensen (2001) is presented, followed by an algorithm for sampling from the posterior and two maximization operations.

2. CG Regressions

Following the notation of Lauritzen and Jensen (2001), a mixed Bayesian network consists of a set of nodes V partitioned into a set of *discrete* nodes, Δ , and a set of continuous nodes, Γ . Each node represents a random variable. The Bayesian network is directed acyclic graph, with the restriction that discrete nodes are not allowed to have continuous parents. Associated with each $Y \in \Gamma$ of the continuous nodes are conditional Gaussian (CG) regressions, one for each configuration in the state space of the discrete parents I of Y , given by

$$\mathcal{L}(Y | I = i, Z = z) = \mathcal{N}(\alpha(i) + \beta(i)^T z, \sigma^2(i)),$$

where $\alpha(i)$ is a real number, Z is a vector of the continuous parents of Y , $\beta(i)$ is a vector of real numbers of the same size of Z , and $\sigma^2(i)$ is a non-negative real number. If the variance $\sigma^2(i) = 0$, then the regression represents a deterministic linear relationship between Y and the Z . Associated with each discrete random variable $X \in \Delta$ is a conditional probability distribution of the variable given its parents in the graph.

The product of the densities associated with the continuous random variables gives the (multivariate normal) density of the continuous variables Γ conditional on the discrete variables Δ . On multiplying this by the product of the conditional probability distributions of each of the discrete variables, the joint density of both discrete and continuous variables is obtained.

Lauritzen and Jensen (2001) introduce as their basic computational object a *CG potential*, represented by the tuple $\phi = [p, A, B, C](H | T)$, where: H is a set of r continuous variables, called the *head*; T is a set of s continuous variables, called the *tail*; $H \cap T = \emptyset$; $p = \{p(i)\}$ is a table of nonnegative numbers; $A = \{A(i)\}$ is a table of $r \times 1$ vectors; $B = \{B(i)\}$ is a table of $r \times s$ matrices; and $C = \{C(i)\}$ is a table of $r \times r$ positive semidefinite symmetric matrices. They introduce various operations on such potentials: multiplication, extension, marginalization, direct combination, complementation, and recursive combination. These operations are required for their message passing algorithm on the junction tree structure, and in the main correspond to operations on probability distributions. However there are restrictions that must be observed for these operations to be permissible; for example, it is not possible to directly combine two CG potentials together if the intersection of their heads is non-empty. Such constraints are obeyed in their propagation algorithm.

In comparison to the propagation scheme presented in this paper, much of the complexity of their algorithm arises because their local computational structure is a strong junction tree of cliques. The cliques with continuous variables essentially contain, after a basic initialization, multivariate CG regressions. Sending a sum-marginal message between two cliques could require marginaliza-

tion over both discrete and continuous variables, in which case the latter are marginalized first. In contrast, we work with univariate regressions on an elimination tree, avoiding matrix operations. (Indeed one could implement the current scheme using the CG potentials and their operations, restricting them to heads that contain only one variable, so that $r = 1$.)

3. Junction Trees and Elimination Trees

In this section we review the notions of a junction tree of cliques and of an elimination tree, and of their relative advantages and disadvantages. More details about these structures may be found in Cowell et al. (1999).

3.1 Making a Junction Tree

Since the work of Lauritzen and Spiegelhalter (1988) the most common graphical structure on which to perform exact inference on Bayesian networks by local message passing has been a junction tree of cliques. This is a tree structure, with the node set being the set of cliques \mathcal{C} of a chordal graph, such that the intersection $C_1 \cap C_2$ of any two cliques $C_1 \in \mathcal{C}$ and $C_2 \in \mathcal{C}$ is contained in every clique on the path in the junction tree between C_1 and C_2 . The intersection of two cliques adjacent in a junction tree is called a *separator*. The basic algorithmic steps in constructing the junction tree, starting from a directed acyclic graph G , are as follows:

1. Add moral edges to G , and then drop directionality on the edges to form the moral graph G^m .
2. Add sufficient fill-in edges to G^m to make a chordal graph G^c .
3. Identify the cliques of G^c , and join them up to form a tree structure which has the running-intersection property.

Step 1 is straightforward, and Step 3 may be done efficiently using the *maximum cardinality search* algorithm (Tarjan and Yannakakis, 1984). It is Step 2, also commonly known as *triangulation*, that presents the main obstacle to efficient message passing. There are many ways to triangulate the moral graph G^m , what is desirable is that the cliques that arise are kept small, or more specifically the sum total state space size over the cliques is minimized. Finding optimal triangulations is NP-hard (Yannakakis, 1981), and so early work focused on heuristic algorithms, typically of a one-step-look-ahead type (Kjærulff, 1990), but other methods, for example genetic algorithms (Larrañaga et al., 1997) have also been used. More recent work has focussed on divide-and-conquer approaches that can yield close to optimal or even optimal triangulations (Becker and Geiger, 2001; Olesen and Madsen, 2002), and an optimal triangulation algorithm is implemented in the commercial expert system HUGIN.¹

3.2 Making an Elimination Tree

Elimination trees were introduced by Cowell (1994) for analysing decision problems, and are described on pages 58–60 of Cowell et al. (1999). An elimination tree is similar to a junction tree, in that it is a tree structure, but with the node set being a subset of the complete subgraphs of a chordal graph (rather than the set of cliques) such that the intersection $C_1 \cap C_2$ of any two nodes

1. The company web site is at <http://www.hugin.com>.

in the elimination tree is contained in every node on the path in the tree between C_1 and C_2 . The subset of complete subgraphs is determined by an elimination sequence, this being an ordering of the nodes of the chordal graph. The basic steps to make an elimination tree starting with the directed acyclic graph G are as follows:

1. Add moral edges to G , and then drop directionality on the edges to form the moral graph G^m .
2. Take an elimination sequence v_k, v_{k-1}, \dots, v_1 of the k nodes (suitably re-numbered) of the moral graph G^m , and use this to form a triangulated graph G^c .
3. For each node v_i associate a so called *cluster set* of nodes consisting of v_i and its neighbours later in the elimination sequence (and hence of lower number), call this set C_i .
4. Join the sets C_1, C_2, \dots, C_k , which has the running-intersection property, together to form a tree.

Step 1 is the same step as for making a junction tree. For Step 2, we first start with the node v_k , and add edges so that it and its neighbours form a complete subgraph (this will make the set C_k in Step 3). Then move onto node v_{k-1} , add edges so that it and its neighbours that occur later in the elimination sequence form a complete subgraph (this will make the set C_{k-1} in Step 3). Repeat this last step for the other nodes in the elimination sequence. There will be k cluster sets, one for each of the k nodes in the graph G , and $C_1 = \{v_1\}$. The choice of elimination sequence will govern the number of fill-ins in the triangulation, and hence the size of the state space of the elimination tree. Reasonable choices can usually be made by a one-step-look-ahead search. Notice that if one knows an optimal triangulation G^c of G^m , then a perfect numbering of the nodes of G^c could be used as an elimination sequence for G^m . Step 4 may be done in time typically linear in k (but possibly as bad as $O(k^2)$), by the simple expedient of finding in each cluster set C_i the first node v_{e_i} , say, that was eliminated after v_i and then joining C_i to C_{e_i} .

3.3 Comparing Elimination and Junction Trees

In an elimination tree, the set of cluster sets contains the set of cliques of the triangulated graph together with some other sets. Hence in terms of storage requirements for potentials on the sets, elimination trees are less efficient than junction trees. Sometimes they can be very bad, as shown with the following example.

Suppose the original graph G or its moral graph G^m is a complete graph of k nodes, each of which represents a binary random variable. Then there are no fill-in edges to be added as G^m is already triangulated, and the junction tree is a single clique containing all k nodes of G and having a total state space of size 2^k . Now consider the elimination tree, made by using an elimination sequence v_k, v_{k-1}, \dots, v_1 . This will yield k cluster sets, with $C_j = \cup_{i=1}^j \{v_i\}$, having a total state space size given by $2 + 2^2 + \dots + 2^k = 2(2^k - 1)$. That is, it requires almost double the storage requirements of the junction tree. Actually things are worse than this, because we have not taken into account the $k - 1$ separators between adjacent clusters which have total state space size $2 + 2^2 + \dots + 2^{k-1} = 2^k - 2$, thus leading to a factor of almost three in the storage requirements. However it should be emphasized that this is a worst case scenario, and in most applications the overhead is a small fraction of the total state space of the corresponding junction tree.

Now suppose that the variables of G are continuous rather than being discrete, with G now representing a multivariate Gaussian distribution. To represent this distribution in the junction tree will require k values for the means of each variable, and a further $k(k+1)/2$ values for the symmetric covariance matrix, making a total of $k(k+3)/2$ values to be stored. In the corresponding elimination tree, as we shall see, only univariate regressions are stored. Hence in C_1 we store two numbers (a mean and a variance), in C_2 we store three numbers (a mean, one coefficient and a variance), . . . , and in C_k we store $k+1$ numbers, (a mean, $k-1$ coefficients, and a variance). Adding all these together we have a total of $2+3+\dots+(k+1) = k(k+3)/2$ values to be stored, the same as for the junction tree. Thus the use of the elimination tree does not introduce duplication in the values to be stored, in contrast to the discrete case. Hence the elimination tree is *almost* as efficient as the junction tree in storage requirements (only almost, because there will be some extra bookkeeping needed to keep track of which variables are in each of the cluster sets).

3.4 Strongly Rooted Trees

For purely discrete Bayesian networks and purely continuous multivariate Gaussian Bayesian networks the junction or elimination trees described above may be used for exact propagation algorithms, with any clique or cluster set being chosen as the root to which messages are collected to and distributed from. For the conditional Gaussian networks in which both discrete and continuous variables appear, Lauritzen (1992) used the notion of a *marked graph* (Leimer, 1989) to define the structure of a *strongly rooted junction tree* in order to have a manageable propagation scheme which handles the asymmetry between the discrete and continuous variables. This structure is retained in Lauritzen and Jensen (2001). Here we use a similar structure based on elimination trees, which we shall call a *strongly rooted elimination tree*. We shall assume without loss of generality that the graph G of the Bayesian network is connected. Then a strongly rooted elimination tree may be formed in the same way as a standard elimination tree provided that, in the elimination sequence used, all of the continuous variables occur before any of the discrete variables. The cluster sets are joined up as before, and the last cluster formed is taken to be the strong root. (If G has more than one connected component, then we form a strong elimination tree for each component; it can then be useful to introduce an empty cluster set connected to each of the strong roots of the individual elimination trees, and make this the strong root of the forest of elimination trees.)

The reason for using a strong elimination tree will become apparent when we discuss the initialization of the tree and propagation on it. For a more efficient computation scheme (from a storage requirement viewpoint) is it convenient to use a tree structure that is intermediate between a junction tree and an elimination tree, a structure which we call a *strong semi-elimination tree*, introduced in the next section.

3.5 From Elimination Tree to Junction Tree

Given an elimination ordering for a graph G , one can construct a triangulated graph G^c , use maximum cardinality search to find the cliques, and then organize the cliques into a junction tree. Alternatively, one could take the elimination tree and remove the redundant cluster sets that are subsets of cliques, by repeated application of the following result due to Leimer (1989) (see also Lemma 2.13 of Lauritzen (1996) or Lemma 4.16 of Cowell et al. (1999)):

Lemma 3.1 *Let C_1, \dots, C_k be a sequence of sets having the running intersection property. Assume that $C_t \subseteq C_p$ for some $t \neq p$ and that p is minimal with this property for fixed t . Then:*

- (i) *If $t > p$, then $C_1, \dots, C_{t-1}, C_{t+1}, \dots, C_k$ has the running intersection property.*
- (ii) *If $t < p$, then $C_1, \dots, C_{t-1}, C_p, C_{t+1}, \dots, C_{p-1}, C_{p+1}, \dots, C_k$ has the running intersection property.*

The preceding Lemma operates on sets, but in the elimination tree we also have links between sets, which will need to be rearranged if a set is deleted. The following algorithm, based on Lemma 3.1 makes a single pass through the cluster sets of an elimination tree to produce a junction tree, it assumes that the elimination tree is connected. It is convenient to make the edges of the elimination tree directed edges, with directions pointing away from the root C_1 . We may then talk of parents (*pa*) and children (*ch*) of the cluster sets in the tree in the obvious manner.

Algorithm 3.2 (Elimination tree to Junction tree)

1. *Initialize:*

- *An ordered list L of the cluster sets C_1, C_2, \dots, C_k derived from the elimination ordering v_k, v_{k-1}, \dots, v_1 having the running intersection property, and joined to form an elimination tree.*
- *An ordered list J , initially empty.*

2. *While L is non-empty do:*

- *Remove the first element C_t from L ;*
- *If C_t is a clique then append it to the end of J , otherwise:*
 - (a) *find $C_p \in ch(C_t)$ such that p is minimized;*
 - (b) *Remove C_p from L ;*
 - (c) *Set $pa(C_p) = pa(C_t)$;*
 - (d) *In each cluster $c \in ch(C_t) \setminus C_p$ replace C_t in $pa(c)$ with C_p ;*
 - (e) *Add the elements of $ch(C_t) \setminus C_p$ to the set $ch(C_p)$;*
 - (f) *Put C_p at the front of L ;*
 - (g) *Discard C_t .*

It is left to the reader to verify that this repeatedly applies Step (ii) of Lemma 3.1, with Steps 2(c)-(e) updating the connections in the tree. When the algorithm terminates the list J contains the cliques in running intersection order, and the parent and child sets of these cliques contain the links required to make a strong junction tree. An example illustrating the steps in Algorithm 3.2 is shown in Figure 1.

Although the message passing algorithms presented below will work on a strong elimination tree, to optimize the storage requirements it is better to work on a *strong semi-elimination tree*. This is a strong elimination tree in which the purely discrete clusters that are subsets of other purely discrete clusters have been removed, with links among the remaining clusters suitably adjusted. Algorithm 3.3 produces a strong semi-elimination tree from a strong elimination tree.

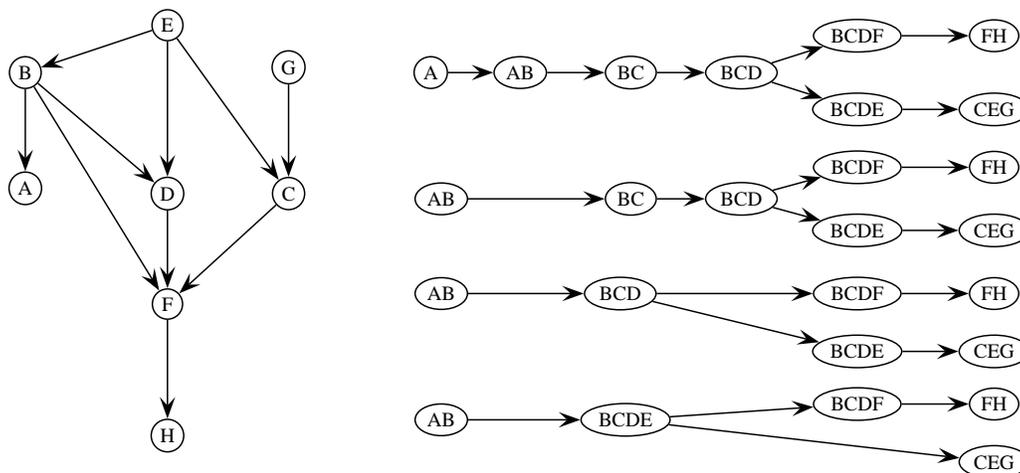


Figure 1: Illustration of Algorithm 3.2. On the left a Bayesian network, and top right the elimination tree obtained using the elimination sequence of reverse alphabetical ordering. In the top tree we first remove the redundant cluster $C_t = \{A\}$ having the unique child cluster $C_p = \{AB\}$, yielding the second tree in which $\{AB\}$ now has no parent. Then the redundant cluster $C_t = \{BC\}$ is removed: it has the unique child cluster $C_p = \{BCD\}$, and so this now inherits $\{BC\}$'s parent $\{AB\}$. Finally the redundant cluster $C_t = \{BCD\}$ is removed. It has two child clusters, of these $C_p = \{BCDE\}$ because E is eliminated after F . C_p has its parent changed to $\{AB\}$ (Step 2b) and is itself made the new parent of $\{BCDF\}$ (Step 2c). It inherits the extra child $\{BCDF\}$ from C_t (Step 2d) to yield the bottom tree, which after dropping directions on the edges gives the junction tree.

Algorithm 3.3 (Strong elimination tree to strong semi-elimination tree)1. *Initialize:*

- An ordered list L of the cluster sets C_1, C_2, \dots, C_k derived from the strong elimination ordering v_k, v_{k-1}, \dots, v_1 having the running intersection property.
- An ordered list J , initially empty.

2. *While L is non-empty do:*

- Remove the first element C_t from L ;
- If C_t contains a continuous variable, or C_t is purely discrete and not a subset of another purely discrete cluster, append it to J , otherwise:
 - Find $C_p \in ch(C_t)$ such that p is minimized; (note that C_p will be purely discrete and $C_t \subset C_p$), and then:
 - (a) Remove C_p from L ;
 - (b) In C_p set $pa(C_p) = pa(C_t)$;
 - (c) In each cluster $c \in ch(C_t) \setminus C_p$ replace C_t in $pa(c)$ with C_p ;
 - (d) Add the elements of $ch(C_t) \setminus C_p$ to the set $ch(C_p)$;
 - (e) Put C_p at the front of L ;
 - (f) Discard C_t .

4. Computations on a Simple Network

The computational scheme to be presented here is more complicated than for the purely discrete case. Although both start with moralizing the Bayesian network, for CG networks a strong triangulation is required, leading to a strong elimination tree. For discrete networks, after the junction tree of cliques has been made, the initialization stage is quite straightforward, consisting of: (i) setting all entries in all of the tables (potentials) in the cliques and separators of the junction tree to unity; (ii) a multiplication of each conditional probability table of the Bayesian network into any one suitable clique table (one whose clique variables contains the variables of the conditional probability table); and (iii) propagation on the junction tree to yield clique and separator tables storing marginal distributions. For the CG networks we use the same initialization process for the discrete part of the elimination tree. However on the continuous part of the tree things are more complicated, and its initialization is perhaps the most important and complicated part of the computational scheme of this paper. In the initialization phase, CG regressions are passed between continuous clusters, and so we introduce a list structure called a *postbag* to store such messages that are to be passed. In addition we introduce for each continuous cluster another list structure called an *lp-potential* that will on completion of the initialization phase store the conditional density of the elimination variable associated with the cluster (conditional on the remaining variables in the cluster). Each regression of the Bayesian network is initially allocated to either some *lp-potential* or *postbag* by rules given below. During the initialization process the contents of the *lp-potential*'s and *postbag*'s may be modified by an operation that we call EXCHANGE (see Section 5.3), which is equivalent to an arc-reversal in the Bayesian network. These EXCHANGE operations have to be done in a correct order, which requires a *postbag*'s contents to be sorted.

Evaluating node marginals is also more complicated. For discrete variables this proceeds as in the purely discrete case, by a marginalization of the tables in the discrete clusters in the elimination tree. For continuous variables we use the PUSH operation introduced by Lauritzen and Jensen (2001), which is a sequence of EXCHANGE operations carried out among regressions in continuous clusters along a path in the tree, to obtain for a continuous variable its marginal density conditional on a set of discrete variables. When combined with the marginal of those discrete variables we obtain the marginal density of the continuous variable—typically a mixture. The PUSH operation is also used as part of the process of entering evidence about continuous variables.

In this section we use a simple example to illustrate the steps of constructing a tree from a Bayesian network, initializing the tree, and finding the marginal of a continuous node, before giving the formulation of these algorithmic steps for a general network in the subsequent section. Some terminology is also introduced that will be used later.

4.1 Specification

Figure 2 shows a small mixed Bayesian network consisting of three discrete random variables, shown as square-shaped nodes, and three continuous real-valued random variables, shown as circles. It follows from the structure of the Bayesian network that the joint distribution of the discrete

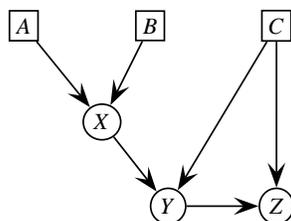


Figure 2: A mixed Bayesian network: square nodes represent discrete variables, circular nodes continuous variables.

variables is $P(A = a, B = b, C = c) = P(A = a)P(B = b)P(C = c)$, where the marginal probability distributions on each individual variable are assumed specified. To complete the probabilistic specification, we require the set of linear regressions (with the α 's, β 's and σ 's being constants):

$$\begin{aligned} \mathcal{L}(X|A = a, B = b) &= \mathcal{N}(\alpha_{X:ab}, \sigma_{X:ab}^2) \\ \mathcal{L}(Y|X = x, C = c) &= \mathcal{N}(\alpha_{Y:c} + \beta_{Y:c}x, \sigma_{Y:c}^2) \\ \mathcal{L}(Z|Y = y, C = c) &= \mathcal{N}(\alpha_{Z:c} + \beta_{Z:c}y, \sigma_{Z:c}^2) \end{aligned}$$

So for example, if all discrete variables are binary, four regressions are required for X , and two each for Y and Z . This set of linear regressions defines the joint density of the continuous variables conditional on the discrete variables.

4.2 Making the Elimination Tree

The first stage is to transform the Bayesian network into the tree on which the message passing takes place. There are 36 possible elimination sequences that could be applied to the moral graph (3!

ways of eliminating the continuous variables first, followed by $3!$ ways of eliminating the discrete variables); here we shall use a sequence in which Y is eliminated first, then X . This is not the most computationally efficient sequence (eliminating Z first will ensure that Z does not appear in a cluster with X), but helps illustrate the operations of the propagation algorithm. In Figure 3 we show various trees that may be formed based on this elimination sequence; for our propagation algorithm we shall use the middle tree, which has the strong root $\{ABC\}$.

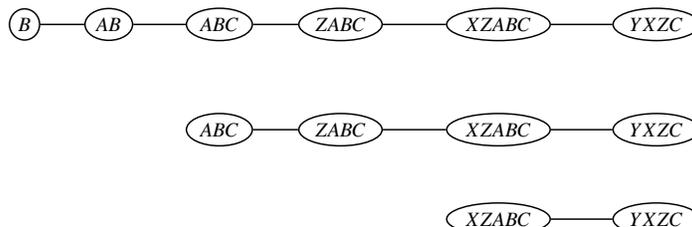


Figure 3: Strong elimination tree (top) with strong root $\{B\}$, strong semi-elimination tree with strong root $\{ABC\}$ (middle) and strong junction tree (bottom) with strong root $\{XZABC\}$, using the elimination sequence $YXZCAB$ applied to the moralized graph derived from the network in Figure 2. Separators are not shown.

4.3 Assignment of Potentials

The next stage is to assign the conditional probabilities and densities of the Bayesian network to the tree so that the tree contains a representation of the joint density of all of the variables. For purely discrete Bayesian networks, and in the formulations of mixed networks of Lauritzen (1992) and Lauritzen and Jensen (2001), each conditional distribution of a discrete node given its parents may be assigned to any clique in the junction tree that contains the family of the node, and similarly for the continuous variables the conditional density of a node given its parents is assigned to any clique containing the family of the node.

In contrast we apply a more restrictive assignment: for continuous variables the conditional density of a node given its parents is assigned to any cluster containing the family of the node, but the conditional distribution of a discrete node given its parents may be assigned to any cluster set that contains the family of the node provided that the cluster set does not contain any continuous variables. There always exists such a cluster set, because discrete variables are not eliminated until after the continuous variables are eliminated.

As a consequence of this assignment, the subtree consisting of those clusters containing only discrete variables contains all of the information required to reconstruct the joint marginal of the discrete variables; this part of the tree shall be referred to as the *discrete part*. Similarly, the clusters having the continuous variables hold representations of all of the densities with which one can construct the joint density of the continuous variables conditional on the discrete variables; this part of the tree will be called the *continuous part*. The set of discrete clusters that are neighbours to the continuous part will be called the *boundary*. Thus in Figure 3 the discrete part of the middle tree consists of the set $\{ABC\}$, the continuous part consists of the sets $\{ZABC\}$, $\{XZABC\}$ and $\{YXZC\}$, and the boundary consists of the single set $\{ABC\}$. We assign probability tables and regressions as

follows:

Cluster	Regressions
ABC	$P(A), P(B), P(C)$
$ZABC$	–
$XZABC$	$\mathcal{L}(X A, B)$
$YXZC$	$\mathcal{L}(Z Y, C), \mathcal{L}(Y X, C)$

Our aim is that when initialization is complete, on the continuous part of the tree we have the following regressions:

Cluster	Regressions
$ZABC$	$\mathcal{L}(Z A, B, C)$
$XZABC$	$\mathcal{L}(X Z, A, B, C)$
$YXZC$	$\mathcal{L}(Y X, Z, C)$

and in general we aim to represent in each continuous cluster set the conditional densities of the eliminated node given the remaining variables in the cluster set. This corresponds to the set-chain representation (Lauritzen and Spiegelhalter, 1988), at least on the continuous part of the tree. Note that we do not explicitly store potentials on separators between continuous cluster sets. In contrast for the discrete part of the tree we will retain separators, and perform usual sum-propagation to find the sum-marginal potential representation (see Cowell et al. (1999), Chapter 6); however in this example there is no such separator.

We move from the initial assignment of regressions to this (partial) set-chain representation by local message passing as follows. First, in the cluster set $\{YXZC\}$ we rearrange the pair of regressions as follows,

$$\mathcal{L}(Z|Y, C = c), \mathcal{L}(Y|X, C = c) \rightarrow \mathcal{L}(Z, Y|X, C = c) \rightarrow \mathcal{L}(Y|Z, X, C = c), \mathcal{L}(Z|X, C = c)$$

for each value that C can take. This rearrangement corresponds to the arc-reversal of the directed edge from Y to Z in Figure 2 (that is, an application of Bayes' theorem). In the expression on the right the regressions $\mathcal{L}(Y|ZXC)$ are retained in the cluster set. The regressions $\mathcal{L}(Z|X, C)$, which are independent of Y , are forwarded to the neighbouring cluster set towards the root, here $\{XZABC\}$. After this rearrangement we have the following represented on the continuous part of the tree:

Cluster	Regressions
$ZABC$	–
$XZABC$	$\mathcal{L}(X A, B), \mathcal{L}(Z X, C)$
$YXZC$	$\mathcal{L}(Y X, Z, C)$

Next, the regressions in the cluster $\{XZABC\}$ are modified as follows:

$$\begin{aligned} &\mathcal{L}(X|A = a, B = b), \mathcal{L}(Z|X, C = c) \\ &\rightarrow \mathcal{L}(X, Z|A = a, B = b, C = c) \\ &\rightarrow \mathcal{L}(X|Z, A = a, B = b, C = c), \mathcal{L}(Z|A = a, B = b, C = c). \end{aligned}$$

The regressions $\mathcal{L}(X|Z, A, B, C)$ are retained in the cluster $\{XZABC\}$, and the regressions $\mathcal{L}(Z|A, B, C)$ are forwarded to the cluster $\{ZABC\}$, and we are done.

Note that it is not necessary to form the intermediate joint density implied by, for example, $\mathcal{L}(X, Z|A, B, C)$. Instead, the algebraic EXCHANGE formulae (see Section 5.3) may be applied to

pass directly from one pair of regressions to another, even in the case that some variances are zero (corresponding to deterministic linear relationship between continuous variables). These formulae are essentially equivalent to Theorem 1 of Shachter and Kenley (1989). By working directly and only with linear regressions, instead of multivariate conditional Gaussian densities, the need for matrix algebra and evaluation of determinants is avoided. Another advantage of the EXCHANGE formulae, as emphasized by Shachter and Kenley (1989), is that they should not lead through roundoff error to negative values of variances, something that could happen when manipulating multivariate densities.²

4.4 Calculating Marginals

One of the prime applications of the propagation algorithms in Bayesian networks is to evaluate marginal distributions of variables, perhaps conditional on values (evidence, findings) of some other variables in the network. We illustrate the procedure for the simple example, beginning with the case that no variable in the network has evidence.

Suppose that we wish to evaluate the marginal density of Z . Formally this may be written as

$$\begin{aligned} f_Z(z) &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} f_{XYZ|ABC}(x,y,z|a,b,c) p(a,b,c) dy dx, \\ &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} f_{XZ|ABC}(x,z|a,b,c) p(a,b,c) dx \\ &= \sum_{a,b,c} f_{Z|ABC}(z|a,b,c) p(a,b,c), \end{aligned}$$

where $f_{XYZ|ABC}(x,y,z|a,b,c)$ is the multivariate normal density of the continuous variables given the values of the discrete variables, etc. Now on our elimination tree we have a representation of $f_{XYZ|ABC}(\cdot)$ in terms of the three linear regressions $\mathcal{L}(Y|X,Z,C)$, $\mathcal{L}(X|Z,A,B,C)$ and $\mathcal{L}(Z|A,B,C)$, and the last of these combined with the joint distribution $P(A,B,C)$ stored in the discrete cluster $\{ABC\}$ is sufficient to evaluate the marginal density of Z , which will thus be a mixture of normal densities.

Now suppose we wish to evaluate the marginal density of Y . Formally this is given by

$$\begin{aligned} f_Y(y) &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} \int_{z=-\infty}^{\infty} f_{XYZ|ABC}(x,y,z|a,b,c) p(a,b,c) dz dx, \\ &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} \int_{z=-\infty}^{\infty} f_{Y|XZC}(y|x,z,c) f_{X|ZABC}(x|z,a,b,c) \times \\ &\quad f_{Z|ABC}(z|a,b,c) p(a,b,c) dz dx, \end{aligned} \tag{1}$$

where in (1) we have written down the factorization of the joint density as available on the tree. We wish to evaluate this by local message passing. To do this we rearrange the current factorization into a more suitable form. First we take the pair of densities $f_{Y|XZC}(y|x,z,c)$, $f_{X|ZABC}(x|z,a,b,c)$ and use the EXCHANGE operation to rewrite these as the pair $f_{X|YZABC}(x|y,z,a,b,c)$, $f_{Y|ZABC}(y|z,a,b,c)$

2. Or so in theory! In implementing the algorithms described in this paper, the author came across the problem that when subtracting one zero double precision from another, the result was zero but with the negative bit set, and so was treated by the compiled program as a negative number! Tracking down this ‘bug’ took the author a couple of days.

which leaves the product density unchanged. Now integrating over X (which gives unity) leaves the following expression to evaluate:

$$f_Y(y) = \sum_{a,b,c} \int_{z=-\infty}^{\infty} f_{Y|ZABC}(y|z, a, b, c) f_{Z|ABC}(z|a, b, c) p(a, b, c) dz,$$

We now apply the EXCHANGE operation again to the two densities in this expression to yield

$$f_Y(y) = \sum_{a,b,c} \int_{z=-\infty}^{\infty} f_{Z|YABC}(z|y, a, b, c) f_{Y|ABC}(y|a, b, c) p(a, b, c) dz,$$

in which the Z integral gives unity, leaving the desired marginal

$$f_Y(y) = \sum_{a,b,c} f_{Y|ABC}(y|a, b, c) p(a, b, c).$$

This sequence may be described in terms of a local message passing as follows: (i) take the regression stored in the cluster $\{YXZC\}$ and pass it to $\{XZABC\}$; (ii) perform the EXCHANGE operation on the pair of regressions stored in $\{XZABC\}$; (iii) take the resulting regression that has Y in the head and pass it to $\{ZABC\}$; (iv) perform the EXCHANGE operation on the pair of regressions now stored in $\{ZABC\}$; (v) combine the resulting regression that has Y in the head with the joint distribution $P(A, B, C)$ that may be obtained from the boundary set to yield the marginal of Y .

This process of rearranging the regressions involving Y is an example of the PUSH operation introduced by Lauritzen and Jensen (2001). (Their PUSH operation can act more generally on a group of variables, but because we are working with an elimination tree we only require it to act on one variable at a time.) We say that the variable Y has been PUSHED to the boundary. The PUSH operation leaves unchanged the overall joint density of the continuous variables conditional on the discrete variables.

Lauritzen and Jensen also used the PUSH operation to incorporate evidence on continuous variables, and we shall follow them. Thus suppose that we wish to evaluate the marginal density of Z given $Y = y^*$. The first step is to PUSH the variable Y to the boundary, and when it arrives there substitute $Y = y^*$ in all cluster sets in which it appears: In the cluster neighbouring the boundary Y appears in the head and the tail is empty of continuous variables, and so this yields a likelihood term for each configuration of the discrete variables that is passed to (that is, multiplied into the discrete potential stored in) the boundary set. The tree then stores the following representations:

Cluster	Regressions and Distributions
ABC	$P(A, B, C, Y = y^*)$
$ZABC$	$\mathcal{L}(Z A, B, C, Y = y^*)$
$XZABC$	$\mathcal{L}(X A, B, C, Z, Y = y^*)$
$YXZC$	—

We may now take the CG regressions stored in $\{ZABC\}$ and combine them with the marginal $P(A, B, C, Y = y^*)$ to obtain, after normalization, the marginal density of Z given the evidence. To obtain the marginal density of X , we would PUSH the regressions in $\{XZABC\}$ to the boundary. Evidence about the discrete variables may also be entered, but only on the discrete part of the tree. For evidence on several continuous variables it is convenient to enter the evidence one variable

at a time. For example, to enter additional evidence that $X = x^*$, we first PUSH the regression $\mathcal{L}(X|A, B, C, Z, Y = y^*)$ to the boundary, and then substitute $X = x^*$, yielding the representation:

Cluster	Regressions and Distributions
ABC	$P(A, B, C, X = x^*, Y = y^*)$
$ZABC$	$\mathcal{L}(Z A, B, C, X = x^*, Y = y^*)$
$XZABC$	—
$YXZC$	—

from which the marginal of Z given the evidence may be found. Notice that the likelihood of the evidence is found, as usual, as the normalization constant when all evidence has been collected to the strong root.

5. The General Local Propagation Scheme

The example in the previous section has illustrated the main steps in initialization and evidence propagation, and evaluation of marginal densities of variables. However the example is too simple to exhibit all of the subtleties involved in the general procedure. In this section we assume that we have a strong elimination tree, and that it is connected. If the tree is disconnected, then the scheme described below may be applied to each connected component separately. The algorithms will also work for a strong semi-elimination tree, and in implementations this might be preferred on efficiency grounds; no details of the following algorithms depend on which of the two types of tree is used.

5.1 Some Notation and Terminology

Let us suppose that the original conditional-Gaussian Bayesian network has $n > 0$ continuous variables $\Gamma = \{\gamma_1, \dots, \gamma_n\}$ and $m > 0$ discrete variables $\Delta = \{\delta_1, \dots, \delta_m\}$. Also suppose that the variables have been numbered so that the elimination ordering of the continuous variables to make the strong elimination tree is $(\gamma_n, \gamma_{n-1}, \dots, \gamma_1)$. The ordering of the discrete variables is unimportant for our purposes, except that it should lead to a computationally efficient tree. We denote the set of continuous cluster sets by \mathcal{C}_Γ , with $C_\Gamma(i) \in \mathcal{C}_\Gamma$ denoting the cluster set associated with eliminating the continuous variable γ_i . Let \mathcal{S}_Γ denote the separators adjacent to continuous cluster sets, with $S_\Gamma(i) \in \mathcal{S}_\Gamma$ denoting the separator between $C_\Gamma(i)$ and its neighbouring cluster set in the direction of the strong root. Note that if the neighbouring cluster is part of the boundary (that is, purely discrete), then $S_\Gamma(i)$ will be purely discrete. We will denote the set of purely discrete clusters by \mathcal{C}_Δ , and the set of separators between purely discrete clusters by \mathcal{S}_Δ .

In the present propagation scheme, the conditional distribution of the continuous variables given the discrete variables is maintained in factored form by sets of univariate regressions. The messages passed between the continuous clusters consists of such sets. To facilitate discussion of this we introduce data structures to store such sets of regressions.

In each continuous cluster we retain, for each configuration of the discrete variables in the cluster, two list structures to store zero or more linear regressions, one that we call the *postbag*, the other we call the *lp-potential* (*lp* is short for *linear predictor*). On the separators \mathcal{S}_Γ we retain only the *postbag*. On the clusters \mathcal{C}_Δ and separators \mathcal{S}_Δ of the discrete part of the tree we store the usual tables (called potentials) of discrete junction tree propagation algorithms.

5.2 Pre-initializing the Tree

After constructing the tree from the Bayesian network, the first task is to allocate the conditional probability tables and CG regressions to the clusters of the tree. We adopt a more restrictive allocation than Lauritzen and Jensen (2001) in that we restrict where the probability tables are allocated. The allocation scheme is as follows:

Algorithm 5.1 Allocation of potentials

Pre-initialize tree potentials

In each continuous cluster set $C_\Gamma(i) \in C_\Gamma$:

- Set each lp-potential to empty;
- Set each postbag to an empty list.

In each $S_\Gamma(i) \in S_\Gamma$: set each postbag to an empty list.

Set all table entries to unity in all potentials of the clusters C_Δ and separators S_Δ of the discrete part of the tree.

Allocation of probability tables

For each $X \in \Delta$: multiply $P(X | pa(X))$ into the potential of any one discrete cluster of C_Δ that contains $X \cup pa(X)$.

Allocation of CG regressions

For each $X \in \Gamma$, find a cluster, $C_\Gamma(i)$ say, which contains $X \cup pa(X)$, and:

- IF the elimination node $\gamma(i)$ of $C_\Gamma(i)$ is X , then add $\mathcal{L}(X | pa(X))$ to the lp-potential;
OTHERWISE append $\mathcal{L}(X | pa(X))$ to the postbag of $C_\Gamma(i)$.

Under this allocation the discrete part of the tree contains all of the conditional (and unconditional) probability tables of the discrete variables, and the product of the potentials in the discrete clusters yields the joint marginal distribution of the discrete variables. In the continuous part of the tree are contained all of the CG regressions of the continuous variables in the Bayesian network, hence the lists in the clusters in the continuous part of the tree represent (in factored form) the joint multivariate density of the continuous variables given the discrete variables.

Applying this to the example of Section 4, both the *postbag* and *lp-potential* of $\{ZABC\}$ were empty, the *postbag* of $\{XZABC\}$ was empty but its *lp-potential* stored $\mathcal{L}(X | A, B)$, whilst for $\{YXZC\}$ the *postbag* contained $\mathcal{L}(Z | Y, C)$ and the *lp-potential* stored $\mathcal{L}(Y | X, C)$.

Note that the continuous leaves of the elimination tree will have non-empty *lp-potentials*, because for each such leaf, its associated elimination node appears nowhere else in the tree, and so there is no other cluster where the conditional density of that variable can be allocated.

5.3 The EXCHANGE Operation

The basic formula required in the message passing scheme of this paper is the EXCHANGE *operation*, which is essentially Bayes' theorem. Let Y, Z, W_1, \dots, W_l be continuous variables (where Y and Z do not refer to the variables in our example), and $a_0, a_1, \dots, a_l, b, c_0, c_1, \dots, c_l, \sigma_Y$ and $\sigma_{Z|Y}$ be constants, with $b \neq 0$, such that we have the pair of CG regressions:

$$\begin{aligned} Z|Y, W_1, \dots, W_l &\sim N(a_0 + a_1 W_1 + \dots + a_l W_l + bY, \sigma_{Z|Y}^2), \\ Y|W_1, \dots, W_l &\sim N(c_0 + c_1 W_1 + \dots + c_l W_l, \sigma_Y^2). \end{aligned}$$

Then the EXCHANGE operator converts these into the pair of distributions

$$Y|Z, W_1, \dots, W_l \sim N(\cdot, \cdot) \text{ and } Z|W_1, \dots, W_l \sim N(\cdot, \cdot), \quad (2)$$

that have the same joint density as the original pair. For convenience we introduce and define $W_0 \equiv 1$. We show in Appendix A that

$$Z|W_1, \dots, W_l \sim N\left(\sum_{i=0}^l (a_i + bc_i)W_i, \sigma_{Z|Y}^2 + b^2\sigma_Y^2\right).$$

For the conditional distribution of Y there are three cases to consider.

Case 1: $\sigma_Y^2 > 0$ and $\sigma_{Z|Y}^2 > 0$:

$$Y|Z, W_1, \dots, W_l \sim N\left(\frac{\sum_{i=0}^l (c_i \sigma_{Z|Y}^2 - a_i b \sigma_Y^2) W_i + b \sigma_Y^2 Z}{\sigma_{Z|Y}^2 + b^2 \sigma_Y^2}, \frac{\sigma_Y^2 \sigma_{Z|Y}^2}{\sigma_{Z|Y}^2 + b^2 \sigma_Y^2}\right).$$

Case 2: $\sigma_Y^2 > 0$ and $\sigma_{Z|Y}^2 = 0$:

$$Y|Z, W_1, \dots, W_l \sim N\left(\frac{Z - \sum_{i=0}^l a_i W_i}{b}, 0\right).$$

Case 3: $\sigma_Y^2 = 0$ and $\sigma_{Z|Y}^2 \geq 0$:

$$Y|Z, W_1, \dots, W_l \sim N\left(\sum_{i=0}^l c_i W_i, 0\right).$$

The derivation of these formulae is straightforward and is given in Appendix A. Case 1 is equivalent to Theorem 1 of Shachter and Kenley (1989), but differing in that they use the central-moment representation of multivariate Gaussian distributions, whereas the EXCHANGE operation (like Lauritzen and Jensen (2001)) employs the raw-moment representation. Cases 2 and 3 may be obtained as mathematical limits of Case 1, however computer implementations would require these to be treated separately.

Finally, if $b = 0$ (so that $\mathcal{L}(Z|Y, W_1, \dots, W_l)$ does not depend on Y), the EXCHANGE operation merely permutes the two regressions, with Y and Z disappearing from the conditioning sets of the two regressions.

5.4 Initial Transformation of the Tree: An Example

Having allocated potentials according to Algorithm 5.1, we proceed to complete the initialization phase via local message passing to yield, on the continuous part of the tree, univariate regressions in the *lp-potentials* and empty *postbags*, so representing, in set-chain form on the continuous part of the tree, the joint density of the continuous variables conditional on the discrete variables. In the example of Section 4 no *postbag* contained more than one regression (for each discrete configuration); in larger and more complicated networks this might not happen, and so within each continuous cluster a sequence of EXCHANGE operations may be necessary. When this happens care must be taken in the ordering of such operations. We illustrate this in an example before giving the general procedure.

The example we shall use is given as Example 3 in Lauritzen and Jensen (2001), and is shown in Figure 4. This example is chosen because Lauritzen and Jensen show that in their initialization phase several layers of recursive combinations of potentials are required. We shall see how this arises and is avoided in our propagation scheme.

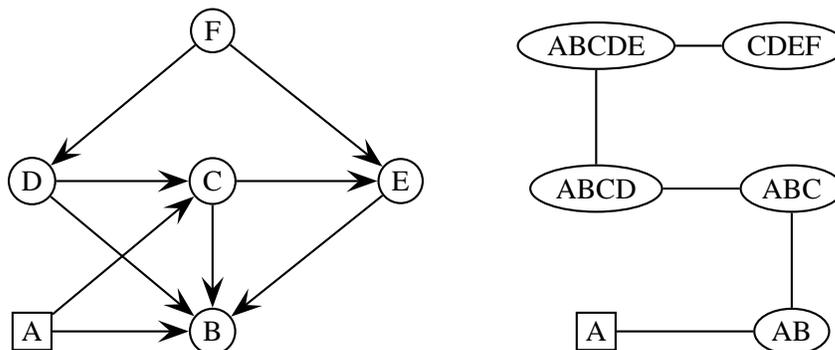


Figure 4: Mixed Bayesian network (left) with one discrete variable A , given as Example 3 in Lauritzen and Jensen (2001). In the strong elimination tree (right) the strong root is a , and the two sets $\{CDEF\}$ and $\{ABCDE\}$ would by themselves form a strong junction tree with $\{ABCDE\}$ as the strong root.

We shall use the same initial allocation of regressions as Lauritzen and Jensen, which means that $\mathcal{L}(F | -)$ is put into the *lp-potential* of cluster $\{CDEF\}$, and in its *postbag* we place $\mathcal{L}(E | CF)$ and $\mathcal{L}(D | F)$. In the *postbag* of cluster $\{ABCDE\}$ we place the regressions $\mathcal{L}(C | DA)$ and $\mathcal{L}(B | ACDE)$. The *lp-potential* of cluster $\{ABCDE\}$ is empty, as are the *lp-potentials* and *postbags* of the three remaining continuous clusters $\{ABCD\}$, $\{ABC\}$ and $\{AB\}$. The discrete distribution $P(A)$ is allocated to the strong root $\{A\}$. This allocation is displayed in the following table.

Cluster	<i>lp-potential</i>	<i>postbag</i>
{CDEF}	$\mathcal{L}(F -)$	$\mathcal{L}(E CF), \mathcal{L}(D F)$
{ABCDE}	-	$\mathcal{L}(C DA), \mathcal{L}(B ACDE)$
{ABCD}	-	-
{ABC}	-	-
{AB}	-	-
{A}	$P(A)$	

Our aim is to transform the *lp-potential* of cluster {CDEF} into $\mathcal{L}(F|CDE)$, (because F is the elimination variable for this cluster) using the original *lp-potential*, the *postbag* contents, and EXCHANGE operations. This corresponds to arc-reversal operations on the two directed edges $F \rightarrow D$ and $F \rightarrow E$ in the original Bayesian network. Now if we first reverse the arc $F \rightarrow E$, then this will give rise to an illegal directed-cycle in the Bayesian network ($E \rightarrow F \rightarrow D \rightarrow C \rightarrow E$); this is avoided if we first reverse the arc $F \rightarrow D$. Hence our sequence of EXCHANGE operations is summarized by

$$\begin{aligned} \underbrace{\mathcal{L}(F|-), \mathcal{L}(D|F), \mathcal{L}(E|CF)} &\rightarrow \mathcal{L}(D|-), \underbrace{\mathcal{L}(F|D), \mathcal{L}(E|CF)} \\ &\rightarrow \mathcal{L}(D|-), \mathcal{L}(E|CD), \mathcal{L}(F|CDE), \end{aligned}$$

in which the pairings of the potentials in each EXCHANGE operation is indicated. We now retain $\mathcal{L}(F|CDE)$ in the *lp-potential* of cluster {CDEF} and pass the regressions $\mathcal{L}(D|-)$ and $\mathcal{L}(E|CD)$ to the cluster {ABCDE}. Now because the variable E is the elimination variable in this cluster, this means that $\mathcal{L}(E|CD)$ is put into the *lp-potential* and $\mathcal{L}(D|-)$ is put into the *postbag*. Actually, because this cluster contains the discrete variable A , there is an *lp-potential* and a *postbag* for each configuration of A . Hence we really put a copy of $\mathcal{L}(E|CD)$ into each such *lp-potential*, and similarly a copy of $\mathcal{L}(D|-)$ into each such *postbag*. This corresponds to a trivial extension of the regressions to $\mathcal{L}(E|ACD) \equiv \mathcal{L}(E|CD)$ and $\mathcal{L}(D|A) \equiv \mathcal{L}(D|-)$. (In the following we shall assume for brevity that we are working with a particular configuration of A , to avoid repeating the phrase “for each configuration of A ”.)

So now in the cluster {ABCDE} we have the following regressions stored:

$$\begin{array}{ll} \textit{lp-potential} & \mathcal{L}(E|ACD) \\ \textit{postbag} & \mathcal{L}(D|A), \mathcal{L}(C|DA), \mathcal{L}(B|ACDE) \end{array}$$

In our desired set-chain representation we wish to end up with $\mathcal{L}(E|ABCD)$ in the *lp-potential*; we may use the following sequence of EXCHANGE operations,

$$\begin{aligned} \underbrace{\mathcal{L}(E|ACD), \mathcal{L}(D|A), \mathcal{L}(C|DA), \mathcal{L}(B|ACDE)} & \\ \rightarrow \mathcal{L}(D|A), \underbrace{\mathcal{L}(E|ACD), \mathcal{L}(C|DA), \mathcal{L}(B|ACDE)} & \\ \rightarrow \mathcal{L}(D|A), \mathcal{L}(C|DA), \underbrace{\mathcal{L}(E|ACD), \mathcal{L}(B|ACDE)} & \\ \rightarrow \mathcal{L}(D|A), \mathcal{L}(C|DA), \mathcal{L}(B|ACD), \mathcal{L}(E|ABCD), & \end{aligned}$$

in which the first two operations merely permute the regressions, (C and D are already conditioning variables of the regressions of E), and only the last is an application of Bayes’ theorem. From this set, $\mathcal{L}(E|ABCD)$ is retained in the *lp-potential* of {ABCDE}, $\mathcal{L}(D|A)$ is put into the *lp-potential* of {ABCD} (because D is the elimination variable of this cluster), and $\mathcal{L}(C|DA)$ and $\mathcal{L}(B|ACD)$ are put into the *postbag* of {ABCD}. Hence in the cluster {ABCD} we have:

$$\begin{array}{ll} \textit{lp-potential} & \mathcal{L}(D|A) \\ \textit{postbag} & \mathcal{L}(B|ACD), \mathcal{L}(C|DA) \end{array}$$

Now D appears in the tails of both regressions in the *postbag*, hence care must be taken. The correct EXCHANGE sequence is:

$$\begin{aligned} \underbrace{\mathcal{L}(D|A), \mathcal{L}(C|DA)}_{}, \mathcal{L}(B|ACD) &\rightarrow \mathcal{L}(C|A), \underbrace{\mathcal{L}(D|AC), \mathcal{L}(B|ACD)}_{}, \\ &\rightarrow \mathcal{L}(C|A), \mathcal{L}(B|AC), \mathcal{L}(D|ABC) \end{aligned}$$

$\mathcal{L}(D|ABC)$ is retained in the cluster $\{ABCD\}$, $\mathcal{L}(C|A)$ is put into the *lp-potential* of the cluster $\{ABC\}$, and $\mathcal{L}(B|AC)$ is put into its *postbag*.

In the cluster $\{ABC\}$ we apply the EXCHANGE operation,

$$\mathcal{L}(C|A), \mathcal{L}(B|AC) \rightarrow \mathcal{L}(B|A), \mathcal{L}(C|AB),$$

retaining $\mathcal{L}(C|AB)$ in the *lp-potential* and putting $\mathcal{L}(B|A)$ into the *lp-potential* of cluster $\{AB\}$, and we are done.

Before proceeding in the next section to give the general initialization algorithm, we need to explain how we choose the ordering exchange operations to be performed when a *postbag* contains more than one regression. The answer is straightforward: one orders the regressions in a *postbag* by a topological ordering in the original Bayesian network of the response (head) variables in the regressions, such that in the ordering all the parents of a variable X appear to the left of X , and all child variables appear to the right of X . This ordering ensures that the sequence of EXCHANGE operations is valid. (It is essentially Proposition 2 of Shachter and Kenley (1989).)

5.5 Initial Transformation of the Tree: General Algorithm

We now present the general algorithm for initializing the CG regressions of the tree. Recall that γ_i is the elimination variable associated with the continuous cluster $C_\Gamma(i)$.

Algorithm 5.2 (Initialization of CG regressions)

1. *Given:*

- A mixed Bayesian network \mathcal{B} with a strong elimination tree initialized according to Algorithm 5.1.
- The elimination sequence $\gamma_n, \gamma_{n-1}, \dots, \gamma_1$ of the continuous nodes.
- A topological ordering TOP of the variables in \mathcal{B} .

2. *Message passing sequence:*

For $i := n$ step -1 until 1 do:

For each configuration δ_i^* of the discrete variables in $C_\Gamma(i)$ do:

- Sort the regressions in the δ_i^* postbag of $C_\Gamma(i)$ so that their head variables occur in the same sequence as in the topological ordering TOP;
- While the δ_i^* postbag of $C_\Gamma(i)$ is not empty do:

- Remove the first regression R in the δ_i^* postbag of $C_\Gamma(i)$;
- If the tail of R contains γ_i , then modify R and the δ_i^* lp-potential of $C_\Gamma(i)$ via the EXCHANGE operation, such that R does not depend on γ_i .
- For each configuration $\delta_{j_i}^* \supseteq \delta_i^*$, of the discrete variables in the cluster $C_\Gamma(j_i)$ neighbouring $C_\Gamma(i)$ in the direction of the strong root, put a copy of R either (1) in the $\delta_{j_i}^*$ lp-potential of $C_\Gamma(j_i)$ if the head of R is the elimination variable of $C_\Gamma(j_i)$, or (2) in the $\delta_{j_i}^*$ postbag of $C_\Gamma(j_i)$ otherwise.
- Discard R .

Note that by the strong nature of the tree, the set of discrete variables of $C_\Gamma(i)$ is contained in the set of discrete variables of $C_\Gamma(j_i)$. Hence for each configuration $\delta_{j_i}^*$ there will be an induced sub-configuration δ_i^* .

On completion of Algorithm 5.2 all of the *postbags* are empty, and the *lp-potentials* contain the desired set-chain CG regressions. To see this, first note that each EXCHANGE operation is a valid application of Bayes' theorem, and does not at any stage alter the joint conditional density that can be reconstructed from the regressions stored in the *lp-potentials* and *postbags*. Secondly, when the outer i -th loop is performed the *lp-potential* of $C_\Gamma(i)$ is not empty, either $C_\Gamma(i)$ is a leaf cluster, in which case it started out non empty; otherwise it will have started out non empty and remained so, or it will have started out empty but then became non-empty because it received a regression from a neighbouring cluster away from the root. (The conditional density of γ_i given its parents must have been placed either in the *lp-potential* of $C_\Gamma(i)$, or in some *postbag* of a cluster on some path from $C_\Gamma(i)$ through clusters further away from the root than $C_\Gamma(i)$, and will arrive, possibly modified via EXCHANGE operations at $C_\Gamma(i)$ through application of the previous steps of the algorithm for higher values of i .)

5.6 Entering Evidence and the PUSH Operation

Evidence on discrete and/or continuous variables may be entered and propagated on the tree, and posterior distributions of individual nodes found. Discrete evidence is entered and propagated in the usual way, *but only on the discrete part of the tree*. To enter continuous evidence, and to find marginal densities of continuous variables, we use the PUSH operation of Lauritzen and Jensen (2001).

It is convenient to enter evidence on the continuous variables one observed variable at a time. In order to keep track of those variables that have already had their evidence entered, in each continuous cluster we retain a boolean variable—called *activeflag*—which is initially set to TRUE before any continuous evidence has been entered. A value of FALSE indicates that evidence on the elimination variable of the cluster has been entered. A TRUE value indicates that evidence concerning the elimination variable may be entered, or that the marginal density of the variable may be calculated.

Recall that the separator $S_\Gamma(i)$ between a continuous cluster set $C_\Gamma(i)$ that is a neighbour to a discrete cluster in C_Δ only contains those discrete variables in $C_\Gamma(i)$. In every such separator we store a table of real values indexed by the states of the discrete variables which we call a *weight* table.

In the algorithm below the cluster neighbouring $C_\Gamma(i)$ in the direction of the strong root will be denoted by *toroot*(i), and is either another continuous cluster, or a purely discrete boundary cluster.

If $\text{toroot}(i)$ is continuous its index is denoted by r_i (so that $r_i \in \{1, 2, \dots, i-1\}$ and $\text{toroot}(i) \equiv C_\Gamma(r_i)$).

We now state the algorithm for entering evidence on a continuous variable γ_j (indexed according to the elimination ordering) which consists of the finding that $\gamma_j = \gamma_j^*$. Basically it performs a sequence of EXCHANGE operations so that in the final regression in which γ_j is the head variable, no continuous variables appear in the tail; the substitution $\gamma_j = \gamma_j^*$ may then be performed in all regressions in which γ_j appears, and where it is the head variable this substitution forms a likelihood to be incorporated into the discrete part of the tree.

Algorithm 5.3 (The PUSH operation: Entering evidence $\gamma_j = \gamma_j^*$)

- **Given:**
 - A tree with elimination sequence $\gamma_n, \gamma_{n-1}, \dots, \gamma_1$ of the continuous variables.
 - The tree initialized according to Algorithm 5.2, and then possibly having had some evidence already entered and propagated—but not on the variable γ_j —according to this algorithm. (Note: all postbags are empty.)
 - Evidence $\gamma_j = \gamma_j^*$ to enter and propagate on the tree.
- **Step 1:** Enter $\gamma_j = \gamma_j^*$ in all regressions in which γ_j is a conditioning variable.
 - For $i := n$ step -1 until $j+1$ do
 - * IF activeflag of $C_\Gamma(i)$ is TRUE and $\gamma_j \in C_\Gamma(i)$ substitute $\gamma_j = \gamma_j^*$ in every lp-potential regression of $C_\Gamma(i)$.
- **Step 2:** Initialize the loop for pushing γ_j towards a boundary cluster.
 - Set $i := j$.
 - For each configuration δ_i^* of the discrete variables of $C_\Gamma(i)$ move the regression in the δ_i^* lp-potential of $C_\Gamma(i)$ into the δ_i^* postbag of $C_\Gamma(i)$.
 - Set activeflag of $C_\Gamma(i)$ to FALSE.
- **Step 3:** Push γ_j towards a boundary cluster.

while $\text{toroot}(i)$ is not a boundary cluster do:

 - For each configuration $\delta_{r_i}^*$ of the discrete variables of $C_\Gamma(r_i)$ (with induced configuration δ_i^* of the discrete variables of $C_\Gamma(i)$) do:
 - * IF activeflag of $\text{toroot}(i)$ is FALSE, then:
 1. Copy the δ_i^* postbag of $C_\Gamma(i)$ into the $\delta_{r_i}^*$ postbag of $C_\Gamma(r_i)$;
 OTHERWISE:
 1. Copy the δ_i^* postbag of $C_\Gamma(i)$ into the $\delta_{r_i}^*$ postbag of $C_\Gamma(r_i)$
 2. Perform the EXCHANGE operation on the $\delta_{r_i}^*$ lp-potential and postbag of $C_\Gamma(r_i)$ (such that the resulting postbag regression has γ_j as head).
 3. Substitute $\gamma_j = \gamma_j^*$ in the $\delta_{r_i}^*$ lp-potential of $C_\Gamma(r_i)$;
 - Discard the content of every postbag of $C_\Gamma(i)$;
 - Set $i := r_i$.

- **Step 4:** Update the discrete part of the tree.
 1. For each configuration δ_i^* of the discrete variables of $C_\Gamma(i)$ substitute $\gamma_j = \gamma_j^*$ into the density of the regression stored in the δ_i^* postbag of $C_\Gamma(i)$ and store the result in the δ_i^* entry of the weight table of $S_\Gamma(i)$.
 2. Multiply the weight table of $S_\Gamma(i)$ into the discrete cluster potential of $\text{toroot}(i)$;
 3. Empty the postbag of $C_\Gamma(i)$.

Substituting $\gamma_j = \gamma_j^*$ into a regression in which γ_j is in the tail, and γ_v is the head variable will result in another regression of γ_v on the remaining variables of the tail, affecting the expression for the mean. In the more complex substitutions of Step 4(1), there is a regression in the δ_i^* postbag of $C_\Gamma(i)$ of the form $\gamma_j \sim N(a_{i,j}, \sigma_{i,j}^2)$, so that the δ_i^* entry in the *weight table* will store

$$w[\delta_i^*] = \frac{\exp\left(-(\gamma_j^* - a_{i,j})^2 / 2\sigma_{i,j}^2\right)}{\sqrt{2\pi\sigma_{i,j}^2}}.$$

If $\sigma_{i,j}^2 = 0$, which could (though not necessarily must) occur if there are logical relationships on the continuous variables, this could be mathematically undefined (Lauritzen and Jensen (2001) provide an example). Should such an event occur an implementation of Algorithm 5.3 should warn the user.

To see how the algorithm works, suppose that we start with no evidence having been entered. In this case the continuous part of the tree stores a representation of the conditional density of the continuous variables given the discrete, $\mathcal{L}(\Gamma|\Delta)$. When evidence $\gamma_j = \gamma_j^*$ is entered on a continuous variable γ_j , the sequence of EXCHANGE operations and substitutions of $\gamma_j = \gamma_j^*$ leads to a factored representation $\mathcal{L}(\Gamma \setminus \{\gamma_j\} | \gamma_j^*, \Delta)$, $\mathcal{L}(\gamma_j | \Delta)$ where the first term is represented in the *lp-potentials* of the continuous clusters in which the *activeflag* is TRUE, and the last term is passed as a likelihood term via a weight table to the discrete part of the tree. After standard evidence propagation on the discrete part of the tree, the latter stores a representation of $P(\Delta | \gamma_j = \gamma_j^*, \delta^*)$ or $P(\Delta, \gamma_j = \gamma_j^*, \delta^*)$ depending on whether or not the discrete potentials have been normalized to unity (δ^* represents discrete evidence). If a second piece of evidence is entered, $\gamma_k = \gamma_k^*$ say, the algorithm leads to the active continuous clusters storing a factored representation of $\mathcal{L}(\Gamma \setminus \{\gamma_j, \gamma_k\} | \gamma_j = \gamma_j^*, \gamma_k = \gamma_k^*, \Delta)$ and the further likelihood factor $\mathcal{L}(\gamma_k = \gamma_k^* | \gamma_j = \gamma_j^*, \Delta)$ being multiplied into the discrete part of the tree. At each stage the *activeflags* which are FALSE in the continuous clusters identify, via the elimination variables of the clusters, those continuous variables about which evidence has been entered, the *lp-potentials* in the other clusters (in which the *activeflags* are TRUE) represent the joint density of the unobserved continuous variables given the discrete variables and the observed continuous variables.

After all continuous evidence has been entered, and evidence on discrete variables has been entered and propagation performed on the discrete part of the tree, the discrete part contains a factored representation of the posterior probability of the unobserved discrete variables given the evidence on all variables.

We illustrate the algorithm by revisiting the example in Section 4. We start with no evidence on the continuous part of the tree, which is initialized thus:

Cluster	<i>lp-potentials</i>
ZABC	$Z abc \sim \mathcal{N}(\alpha_{Z:abc}, \sigma_{Z:abc}^2)$
XZABC	$X zabc \sim \mathcal{N}(\alpha_{X:abc} + \beta_{X:Zabc}z, \sigma_{X:abc}^2)$
YXZC	$Y xzc \sim \mathcal{N}(\alpha_{Y:c} + \beta_{Y:Xc}x + \beta_{Y:Zc}z, \sigma_{Y:c}^2)$

If $X = x^*$ is observed, then Algorithm 5.3 operates as follows:

1. In the cluster $\{YXZC\}$, the distribution $Y|xzc \sim \mathcal{N}(\alpha_{Y:c} + \beta_{Y:Xc}x + \beta_{Y:Zc}z, \sigma_{Y:c}^2) \rightarrow Y|x^*zc \sim \mathcal{N}(\alpha_{Y:c} + \beta_{Y:Xc}x^* + \beta_{Y:Zc}z, \sigma_{Y:c}^2)$. The *activeflag* remains TRUE.
2. In $\{XZABC\}$, the *activeflag* is set to FALSE, and for each configuration abc the regression $\mathcal{N}(\alpha_{X:abc} + \beta_{X:Zabc}z, \sigma_{X:abc}^2)$ is moved into the *abc postbag* of $\{ZABC\}$
3. In $\{ZABC\}$, for each configuration abc the EXCHANGE operations acts on the *abc lp-potential*, representing $\mathcal{L}(Z|A = a, B = b, C = c)$, and the *abc postbag* to give a new *abc lp-potential* representing $\mathcal{L}(Z|X = x, A = a, B = b, C = c)$ and modified *abc postbag* content representing $\mathcal{L}(X = x|A = a, B = b, C = c)$. Then the *lp-potential* is set to $\mathcal{L}(Z|X = x^*, A = a, B = b, C = c)$, and the weight table entry $w(a, b, c)$ stores the density value of $\mathcal{L}(X = x^*|A = a, B = b, C = c)$.
4. The *postbags* of $\{ZABC\}$ are emptied, the *activeflag* remains TRUE.
5. In the discrete cluster $\{ABC\}$, the potential $p(a, b, c) \rightarrow p(a, b, c)w(a, b, c) \forall \{a, b, c\}$.
6. On normalizing the potential in the discrete cluster $\{ABC\}$ we obtain the probability distribution $P(A, B, C|X = x^*)$.

5.7 Evaluating Posterior Marginals of Individual Variables

After propagating evidence on variables as described above, finding the posterior marginal of a discrete variable, D say, proceeds in the usual way: Find a cluster set in the discrete part of the tree containing D and marginalize the joint table in that cluster set appropriately.

Finding the posterior density of an unobserved continuous variable uses the PUSH operation in a way similar to but simpler than Algorithm 5.3. Suppose the marginal density of $Y \in \Gamma$ is required. The idea is to use a sequence of EXCHANGE operations to push Y to a cluster C neighbouring the boundary, so that we have a representation of the distribution $\mathcal{L}(Y|\mathcal{E}_\Gamma, B)$ where \mathcal{E}_Γ denotes the evidence on the continuous variables, and $B \subseteq \Delta$ are the discrete variables in the cluster C . From the boundary cluster the marginal $P(B|\mathcal{E}_\Gamma \cup \Delta)$ may be found and then combined with $\mathcal{L}(Y|\mathcal{E}_\Gamma, B)$ to give the required posterior marginal of Y . The complete algorithm is given in Algorithm 5.4, which uses the same notation as Algorithm 5.3.

Algorithm 5.4 (Find the posterior density of a continuous variable)

- *Given:*
 - A strong elimination tree with elimination sequence $\gamma_n, \gamma_{n-1}, \dots, \gamma_1$ of the continuous variables.

- The tree initialized according to Algorithm 5.2, with evidence \mathcal{E}_Γ entered as in Algorithm 5.3, and discrete evidence \mathcal{E}_Δ entered and propagated on the discrete part, so that the discrete clusters contain posterior distributions. (Note: all postbags are empty.)
 - Task: to find the posterior density of an unobserved continuous variable γ_j . (Note: the activeflag of $C_\Gamma(j)$ is TRUE.)
- **Step 1:** Initialize the loop.
 - For each configuration δ_j^* of the discrete variables of $C_\Gamma(j)$ copy the δ_j^* lp-potential of $C_\Gamma(j)$ into the δ_j^* postbag of $C_\Gamma(j)$.
 - Set $i := j$
 - **Step 2:** Push γ_j towards a boundary cluster.

while toroot(i) is not a boundary cluster do:

 - For each configuration $\delta_{r_i}^*$ of the discrete variables of $C_\Gamma(r_i)$ (with induced configuration δ_i^* of the discrete variables of $C_\Gamma(i)$) do:
 - * Copy the δ_i^* postbag of $C_\Gamma(i)$ into the $\delta_{r_i}^*$ postbag of $C_\Gamma(r_i)$;
 - * IF activeflag of $C_\Gamma(r_i)$ is TRUE and γ_j is in the tail of the regression in the $\delta_{r_i}^*$ lp-potential of $C_\Gamma(r_i)$, THEN use the EXCHANGE operation formulae to modify the $\delta_{r_i}^*$ postbag of $C_\Gamma(r_i)$ (so that γ_j is still the head variable) but do not modify the $\delta_{r_i}^*$ lp-potential of $C_\Gamma(r_i)$;
 - Empty all of the postbags of $C_\Gamma(i)$;
 - Set $i := r_i$.
 - **Step 3:** Find the marginal density.
 1. Marginalize the discrete potential in the boundary cluster neighbouring $C_\Gamma(i)$ to the weight table in the separator $S_\Gamma(i)$.
 2. Output the result of adding together the product of each weight table entry with the density of the regression stored in the corresponding postbag of $C_\Gamma(i)$.
 3. Empty all of the postbags of $C_\Gamma(i)$.

Prior to an application of this algorithm the *lp-potentials* in the active clusters represent a factorization of the joint conditional density of the unobserved continuous variables given the evidence on the continuous variables. The algorithm does not change these in any way, and so does not alter this joint conditional density, and indeed the algorithm leaves the tree ready in a state for finding the marginal density of another continuous variable. The algorithm is just using the *postbags* as temporary storage to find, by repeated (partial) application of the EXCHANGE formulae, the marginal density of γ_j conditional on the discrete variables and the values of the observed continuous variables. Step 3 combines this with the correct posterior probability of the unobserved discrete variables of $C_\Gamma(i)$ (conditional on all evidence) to form the posterior marginal density of γ_j .

6. Comparison to Other Methods

In this section we review the propagation scheme described in this paper, and compare it to the scheme of Lauritzen and Jensen (2001) and to the work of Shachter and Kenley (1989). We then discuss some possible extensions of the scheme.

6.1 Summary of Current Scheme

In the current scheme, evidence propagation and evaluation of marginal distributions in a mixed Bayesian network \mathcal{B} takes place on a strong elimination tree or strong semi-elimination tree. The tree has two distinct parts, the continuous part and the discrete part, with the strong root located in the discrete part. The continuous part is initialized using the CG regressions of \mathcal{B} , and represents, using a collection of univariate regressions, the density of the continuous variables conditional on the discrete variables. The discrete part represents the marginal distribution of the discrete variables, and is initialized using the discrete conditional probability tables of \mathcal{B} . Entering evidence on continuous variables, and evaluating marginal densities of continuous variables, uses the PUSH operation with the EXCHANGE formulae, the latter being an application Bayes' theorem. Discrete evidence is entered on the discrete part of the tree and propagated on the discrete part of the tree in the usual way. For finding marginals of continuous variables there is no DISTRIBUTE operation on the tree.

6.2 Comparison with the Lauritzen and Jensen (2001) Scheme

As discussed in Section 2 the Lauritzen and Jensen propagation scheme uses a strong junction tree architecture. Associated with each clique of the junction tree is a CG potential, which is a tuple $\phi = [p, A, B, C](H|T)$ of scalars, vectors and matrices and a partition of the variables into conditioned variables (the head) and conditioning variables (the tail). The conditional distribution or density of a variable X in \mathcal{B} may be assigned to any clique or separator that contains the family of X in the Bayesian network. CG potentials may be combined but there are restrictions that have to be followed, which necessitate the introduction of the *recursive combination* of potentials to allow incoming messages to a clique to be combined correctly.

It is instructive to see how recursive combination is avoided in the current scheme, or alternatively, how to interpret recursive combination within the current scheme. For this we return to the example in Section 5.4. In Figure 4 the clusters $\{CDEF\}$ and $\{ABCDE\}$ form a strong junction tree with the latter as the strong root. In the Lauritzen and Jensen analysis, the assignment of potential to clique $\{CDEF\}$ leads to a CG potential having the head and tail structure $(DEF|C)$. This is decomposed into $(F|CED)$ and $(DE|C)$ the latter is passed to the clique $\{ABCDE\}$ to be combined with the potential $(BC|DE)$. However the heads' and tails' contents of these two potentials preclude their direct combination. Instead they must be combined recursively. The first stage is to decompose $(DE|C) \rightarrow (E|CD), (D|-)$, however this is not sufficient, as $(E|CD)$ cannot be directly combined with $(BC|DE)$. So we decompose $(BC|DE) \rightarrow (B|CDE), (C|D)$ and then we may combine the four potentials $(B|CDE)(E|CD)(C|D)(D|-)$ in that order to yield a potential $(BCDE|-)$.

If one compares these four potentials with the regressions stored in the cluster $\{ABCDE\}$ in Section 5.4 we see that they have the same head-and-tail structures. In the current scheme the regressions $\mathcal{L}(D|-), \mathcal{L}(E|CD)$ were passed to the cluster $\{ABCDE\}$, which stored (omitting the dependence on A) the regressions $\mathcal{L}(C|D), \mathcal{L}(B|CDE)$. These are subject to EXCHANGE operation,

dependent on the topological ordering of the head variables in the Bayesian network \mathcal{B} , the ordering being $D - C - E - B$.

Thus we interpret the recursive combination of potentials as a ordered factorization of potentials so their direct combinations are well defined (although it will not always decompose potentials to univariate CG potentials). The current scheme avoids the recursive combination operation because it works all the time with a factored representation, and the correct ordering of EXCHANGE operations is ensured by using the topological ordering of the variables in \mathcal{B} . It has echoes of *lazy-propagation* (Madsen and Jensen, 1998), in which potentials are stored in factored form when initializing a junction tree and are only combined when required; the difference is that in our scheme factored forms are always retained.

In our scheme there is no SUM-DISTRIBUTE operation on the continuous part of the tree, whereas Lauritzen and Jensen have such an operation that can be used to store weak-marginals in the separators. If weak-marginals are desired in the current scheme, they may be found using the mean and variance of the mixture marginals.

Another aspect of the Lauritzen and Jensen scheme should be mentioned, which is their operation of *minimization* of the tails of CG potentials. In their operations on CG potentials represented by tuples $\phi = [p, A, B, C](H | T)$, when a column of B has entries all zero for every configuration of the discrete variables in the CG potential, then that column and the associated continuous tail variable may be removed. This is a *reduction* operation, and takes place during recursive combination. In the current scheme, reduction occurs as a result of the EXCHANGE operation: the Z regression of (2) does not depend on Y , so an implementation of the EXCHANGE operation would, in taking this into account, automatically perform a reduction. The *minimization* of a potential occurs if the potential has been reduced as far as possible.

Lauritzen and Jensen also discuss the possibility of forming the marginal of a group of continuous variables. This should be possible within the present scheme, with the result being expressed as weighted sets of linear regressions. However it may be that in the message passing process more than one regression might be stored in a *postbag* (for a given configuration of discrete variables) and if so their order would be important, not however the topological ordering of the original Bayesian network variables used in Algorithm 5.2, but the (reverse of) the elimination ordering. This would be appropriate because it is a perfect numbering of the strongly triangulated graph associated with the tree. Similar considerations suggests it ought to be possible to propagate evidence on several continuous variables simultaneously. These connections are discussed in further detail in the next subsection.

6.3 Relationship to the Shachter and Kenley (1989) Scheme

In the Shachter and Kenley scheme, arc-operations are performed on a Bayesian network one pair of variables at a time, which means that it operates on pairs of linear regressions, which is like the current scheme. (Their paper is concerned with pure Gaussian networks, but this is difference is not very significant.) When several arcs need reversing in a Probabilistic Node Reduction operation (their PROPOSITION 2) the sequence of arc reversals has to follow an ordering which is equivalent to one obtained from a topological ordering of the nodes in the Bayesian network. Hence this is very similar to the sequence of EXCHANGE operations in initializing the tree described in Algorithm 5.2.

The close connections between the current scheme, and of both Shachter and Kenley (1989) and Lauritzen and Jensen (2001) are illuminated by the paper of Shachter et al. (1990). These

authors show that the inference algorithms operating on junction trees are essentially the same as node reductions algorithms operating on influence diagrams, because they are both working on the same underlying graphical structures. In the terms of the present paper, what they show is that in the elimination sequence $\gamma_n, \gamma_{n-1}, \dots, \gamma_1$ operating on the moralized mixed Bayesian network, the cluster $C_\Gamma(i)$ is the same as that which would be obtained in the influence diagram for family of the node γ_i after all outgoing arcs have been reversed to make γ_i a barren node, if the nodes are removed from the mixed Bayesian network in the same order as the elimination sequence. To avoid directed cycles being introduced there is a restriction to the order in reversing the child arcs of a node, the restriction uses a topological ordering of the original Bayesian network. They also describe how evidence is entered—if evidence is entered on a node then arcs are reversed so that that node has no parents, and in the process child arcs from the node are removed. When this is done the likelihood associated with that node may be found. This process—which they call *evidence propagation*—is essentially the PUSH operation on a continuous variable, except that arcs are reversed only to the point that the variable has no continuous parents, and then substituting the evidence value leads to modification of those regressions in which the variable appears in the tail (corresponding to removal of those arcs from the influence diagram viewpoint) and evaluation of a likelihood to be passed on to the discrete part of the tree. The elimination tree, with its *lp-potential* and *postbag* structures, provides an organizing framework for such operations. They also describe how multiple evidence may be entered simultaneously, and their procedure should be transferable into the current scheme.

6.4 Implementation Issues

The propagation scheme presented here works by manipulating linear regressions. To aid the presentation the paper has used, like Lauritzen and Jensen (2001), a raw moment representation, for which the EXCHANGE formulae of Section 5.3 may be used. If the user wishes to implement the current scheme then one possibility would be to represent CG regressions by the tuple $\phi = [p, A, B, C](H | T)$ of Lauritzen and Jensen, but now A and C are scalars and B a vector (corresponding to $r = 1$) for each configuration of the discrete variables. Further restrictions are that the table p is a table of 1's if the CG regression has continuous variables, and hence are not required.

However this is not the only possibility. In the author's C++ implementation, an associative array of the form `std::map<variable, double>` is used to store coefficients of covariates in the regressions. The reduction operation is effected by a variable being removed from the associative array.

One could instead use the central-moment representation of Shachter and Kenley, all that would be required would be suitable replacements of the EXCHANGE formulae which describe Bayes' theorem. In this case the formulae in Theorem 1 of Shachter and Kenley could be used (suitably extended to take account of deterministic relationships and the configurations of the discrete variables).

A further possibility could be to use computer algebra. Each univariate regression is specified by (1) a specification of the head and tail variables and either (2a) a quadratic form in the continuous variables if the variance is strictly positive, or (2b) a linear form for deterministic relationships. These are readily represented and manipulated in computer algebra packages, and so the current scheme could be implemented in which the messages are either linear expressions or quadratic forms in the continuous variables. Reduction operations would be taken care of automatically by such computer algebra calculations.

From these comments above we see that, although the presentation in this paper has focussed on using the raw-moment representation, the propagation scheme is more general than this. The only place that the raw-moment representation has been used is in the explicit EXCHANGE formulae of Section 5.3.

7. Sampling and Mode-Finding Algorithms

Aside from finding the posterior marginals of variables, the use of the elimination tree in the current scheme facilitates other operations that may be of interest in applications. In Section 7.1 we show how to sample from the posterior distribution, and in Section 7.2 we show how to find the highest peak in the posterior mixture distribution, which could be useful as the starting point of an iterative search for the posterior mode.

7.1 Sampling from the Posterior

It may be desirable to sample from the posterior distribution of the variables in the Bayesian network given some observed evidence $\mathcal{E}_{\Gamma \cup \Delta}$. Dawid (1992) has shown how to do this for discrete networks, his method is as follows. Starting from a junction tree of cliques, and after entering and sum-propagating evidence \mathcal{E}_{Δ} , the clique and separator tables contain posterior marginals of their variables. Suppose we label the cliques in running intersection order C_1, C_2, \dots, C_k say, with C_1 being chosen as the root clique. First one samples from the posterior marginal in C_1 , to give some configuration δ_1^s . Then one samples from the posterior marginal of the variables in C_2 conditional on δ_1^s , yielding some combined configuration $\delta_1^s \cup \delta_2^s$. Then one samples the variables in C_3 conditional on $\delta_1^s \cup \delta_2^s$. Proceeding in this way will yield a sample $\delta^s = \delta_1^s \cup \delta_2^s \cup \dots \cup \delta_k^s$ from the posterior distribution on the junction tree.

Here we present in Algorithm 7.1 a simple extension of Dawid's method to conditional-Gaussian networks. The idea is to sample the discrete variables, and then sample the remaining continuous variables one at a time in a distribute-type operation.

Algorithm 7.1 (Sampling from the posterior)

- **Given:** A tree with elimination sequence $\gamma_n, \gamma_{n-1}, \dots, \gamma_1$ of the continuous variables, which has been initialized and has had evidence propagated according to Algorithm 5.3, and any discrete evidence has also been propagated on the discrete part of the tree.
- **Sample:**
 - Sample a configuration δ^s of the discrete variables Δ using the algorithm of Dawid.
 - For $i := 1$ step 1 until n do
 - * IF activeflag of $C_{\Gamma}(i)$ is TRUE then find the sub-configuration $\delta_i^s \subseteq \delta^s$ of the discrete variables in $C_{\Gamma}(i)$, and sample γ_i from the regression stored in the δ_i^s lp-potential of $C_{\Gamma}(i)$ in which the sampled or observed values of all continuous tail variables $\in \{\gamma_1^s, \gamma_2^s, \dots, \gamma_{i-1}^s\}$ have been substituted; denote the sampled value by γ_i^s .
 - * OTHERWISE there is evidence $\gamma_i = \gamma_i^*$, so simply set $\gamma_i^s = \gamma_i^*$.
- The configuration $\{\gamma_i^s : i = 1, \dots, n\} \cup \delta^s$ is a sample from the posterior density.

Note that, because of the elimination tree structure employed, each simulated γ_i is sampled from a univariate normal distribution. Such simulation may be done efficiently by a variety of methods and is much simpler than sampling from a multivariate normal distribution.

7.2 Locating the Mode (Approximately)

It is sometimes of interest to locate the most probable values of the unobserved variables given evidence on observed variables. For discrete networks this may be found by local propagation as shown by Dawid (1992). For CG networks an exact solution to the problem is not known, and we do not propose a solution here. Instead in Algorithm 7.2 we propose a method that finds the component having the *highest peak* in the posterior distribution. The posterior distribution is a mixture of weighted multivariate normal densities. Each component multivariate density will attain maximum height at its mean, the height will be proportional to the weight of the component divided by the square root of the determinant of the covariance matrix of the component. These heights are compared by Algorithm 7.2. Now if the variances of the components in the mixture are small compared to the distances between their means, then the location of the component having the highest peak might be expected to be a good approximation to the posterior mode, or could be used as the starting point of an iterative search for the mode. Algorithm 7.2 is slightly more complicated than Algorithm 7.1, and cannot be used if any variances are zero. In order to keep track of the heights of each component, we need to keep a *weight* table in every continuous separator in S_Γ . Note that it is not necessary to evaluate a determinant.

Algorithm 7.2 (Highest Component Search)

- **Given:**

- A tree with elimination sequence $\gamma_n, \gamma_{n-1}, \dots, \gamma_1$ of the continuous variables, which has been initialized and had evidence propagated according to Algorithm 5.3, and any discrete evidence also been propagated on the discrete part of the tree.
- All entries in all weight tables initialized to unity.

- **Highest Peak Search:**

- For $i := n$ step -1 until 1 do
 1. IF activeflag of $C_\Gamma(i)$ is TRUE THEN: For each configuration δ_i^* of the discrete variables in $C_\Gamma(i)$ multiply the δ_i^* entry in the weight table of $S_\Gamma(i)$ by $1/\sqrt{2\pi\sigma^2(\delta_i^*)}$ where $\sigma^2(\delta_i^*)$ is the variance in the regression stored in the δ_i^* lp-potential of $C_\Gamma(i)$.
 2. IF toroot(i) is NOT a boundary cluster, THEN: For each configuration $\delta_{r_i}^*$ of the discrete variables of $C_\Gamma(r_i)$ (with induced configuration δ_i^* of the discrete variables of $C_\Gamma(i)$), multiply the $\delta_{r_i}^*$ entry in the weight table of $S_\Gamma(r_i)$ by the δ_i^* entry in the weight table of $S_\Gamma(i)$.
OTHERWISE if toroot(i) IS a boundary cluster, then multiply the weight table of $S_\Gamma(i)$ into the discrete potential of toroot(i) in the usual way.
- Use the MAX-PROPAGATE algorithm of Dawid (1992) on the discrete part of the tree to give a “max-configuration” δ^m of the discrete variables.
- For $i := 1$ step 1 until n do

- * IF activeflag of $C_\Gamma(i)$ is FALSE, then there is evidence $\gamma_i = \gamma_i^*$, so set the peak value $\gamma_i^m = \gamma_i^*$;
 OTHERWISE activeflag of $C_\Gamma(i)$ is TRUE, so find the sub-configuration $\delta_i^m \subseteq \delta^m$ of the discrete variables in $C_\Gamma(i)$, and set γ_i to the mean of the regression stored in the δ_i^m lp-potential of $C_\Gamma(i)$ in which the peak values or observed values of all continuous tail variables $\in \{\gamma_1^m, \gamma_2^m, \dots, \gamma_{i-1}^m\}$ have been substituted; denote the peak value by γ_i^m .

- The configuration $\{\gamma_i^m : i = 1, \dots, n\} \cup \delta^m$ specifies the location of the highest component in the joint multivariate posterior density.

Mathematically Algorithm 7.2 may be understood in the following manner. Let $I_\Gamma = \{I_1, I_2, \dots, I_k\}$ denote the set of indices of the continuous nodes for which no evidence has been entered (with $I_k > I_{k-1} > \dots > I_1$). Then the algorithm starts out with the tree storing a recursive factorization of the posterior multivariate normal mixture density in the form:

$$p(\Delta | \mathcal{E}_{\Gamma \cup \Delta}) \prod_{i \in I_\Gamma} f_i(\gamma_i | S_\gamma(i), \mathcal{E}_\Gamma)$$

where the $f_i(\cdot | \cdot)$ are appropriate CG regression densities. Given the covariates and evidence, each CG regression density is maximized at the mean, so the component having the maximum height may be obtained as a sequence of ordered maximizations:

$$\begin{aligned} & \max_{\Gamma, \Delta} \left(p(\Delta | \mathcal{E}_{\Gamma \cup \Delta}) \prod_{j=1}^k f_{I_j}(\gamma_{I_j} | S_\gamma(I_j), \mathcal{E}_\Gamma) \right) \\ &= \max_{\Gamma, \Delta} \left(p(\Delta | \mathcal{E}_{\Gamma \cup \Delta}) w_{I_k}(S_\gamma(I_k), \mathcal{E}_\Gamma) \prod_{j=1}^{k-1} f_{I_j}(\gamma_{I_j} | S_\gamma(I_j), \mathcal{E}_\Gamma) \right) \\ &= \max_{\Gamma, \Delta} \left(p(\Delta | \mathcal{E}_{\Gamma \cup \Delta}) w_{I_k}(S_\gamma(I_k), \mathcal{E}_\Gamma) w_{I_{k-1}}(S_\gamma(I_{k-1}), \mathcal{E}_\Gamma) \prod_{j=1}^{k-2} f_{I_j}(\gamma_{I_j} | S_\gamma(I_j), \mathcal{E}_\Gamma) \right) \\ & \vdots \\ &= \max_{\Delta} \left(p(\Delta | \mathcal{E}_{\Gamma \cup \Delta}) \prod_{j=1}^k w_{I_j}(S_\gamma(I_j), \mathcal{E}_\Gamma) \right), \end{aligned}$$

where the $w_i(\cdot)$ are the *weight* tables representing the values of the densities of the CG regressions located at the means. The algorithm accumulates the product of these values and multiplies them into the discrete part of the tree, from which standard MAX-PROPAGATION may be used to find δ^m . (Note that this accumulated product is valid because of the strong nature of the tree: a continuous cluster $C_\Gamma(r_i)$ neighbouring another continuous cluster $C_\Gamma(i)$ but closer to the strong root will contain all of the discrete variables that are in $C_\Gamma(i)$.) This information is then distributed back to locate the mean values of the continuous variables for the configuration δ^m of the discrete variables, in the final stage of Algorithm 7.2.

Finally, we mention another maximization operation called SEMIMAX-PROPAGATION that can easily be carried out in the current scheme. This consists of finding the most likely configuration

of the posterior marginal distribution of the discrete variables, that is, finding the maximum of $P(\Delta | \mathcal{E}_{T \cup \Delta})$. For this we propagate evidence as in Algorithm 5.3, and incorporate evidence on the discrete variables on the discrete part of the tree. We then perform standard MAX-PROPAGATION on the discrete part of the tree according to the algorithm of Dawid (1992). SEMIMAX-PROPAGATION was introduced and applied to forensic DNA problems involving the modelling and analysis of mixed DNA samples using CG networks by Cowell et al. (2004).

8. Summary

We have presented a local propagation scheme for conditional Gaussian Bayesian networks based on elimination trees, that combines the scheme of Lauritzen and Jensen (2001) with that of Shachter and Kenley (1989). Complex matrix algebra is avoided because operations manipulate linear regressions. The propagation scheme is not dependent on a particular implementation of the representation of linear regressions, although the paper has used one for exposition.³ We have also introduced: an algorithm for sampling on such networks; an algorithm for finding highest peaks that could be useful either as an approximation to, or an iterative algorithm for locating, the posterior mode of the distribution; and have briefly described another operation called SEMIMAX-PROPAGATION.

Acknowledgments

The author would like to thank anonymous referees for their very helpful comments and suggestions for making improvements to this paper.

Appendix A. Derivation of EXCHANGE Formulae of Section 5.3

From the pair of normal distributions

$$\begin{aligned} Z | Y, W_1, \dots, W_l &\sim N(a_0 + a_1 W_1 + \dots + a_l W_l + bY, \sigma_{Z|Y}^2), \\ Y | W_1, \dots, W_l &\sim N(c_0 + c_1 W_1 + \dots + c_l W_l, \sigma_Y^2), \end{aligned}$$

it follows that $Y | Z, W_1, \dots, W_l$ and $Z | W_1, \dots, W_l$ are also normal distributions. The mean and variance of the latter is readily found using

$$\begin{aligned} E[Z | W_1, \dots, W_l] &= E[E[Z | Y, W_1, \dots, W_l]] = E[a_0 + a_1 W_1 + \dots + a_l W_l + bY] \\ &= \sum_{i=0}^l (a_i + bc_i) W_i \\ V[Z | W_1, \dots, W_l] &= E[V[Z | Y, W_1, \dots, W_l]] + V[E[Z | Y, W_1, \dots, W_l]] \\ &= E[\sigma_{Z|Y}^2] + V[a_0 + a_1 W_1 + \dots + a_l W_l + bY] \\ &= \sigma_{Z|Y}^2 + b^2 \sigma_Y^2 \end{aligned}$$

where we define $W_0 \equiv 1$.

There are three cases to consider in finding the conditional distribution of $Y | Z, W_1, \dots, W_l$.

3. It is also one implemented by the author.

Case 1: $\sigma_Y > 0$ and $\sigma_{Z|Y}^2 > 0$.

The joint conditional density of $Y, Z | W_1, \dots, W_l$ is

$$\begin{aligned} f_{Y,Z|W_1,\dots,W_l}(y,z) &= f_{Z|Y,W_1,\dots,W_l}(y,z) f_{Y|W_1,\dots,W_l}(y) \\ &= \frac{1}{2\pi\sigma_{Z|Y}\sigma_Y} \exp\left(\frac{-(z - \sum_{i=0}^l a_i W_i - by)^2}{2\sigma_{Z|Y}^2}\right) \exp\left(\frac{-(y - \sum_{i=0}^l c_i W_i)^2}{2\sigma_Y^2}\right) \\ &= f_{Y|Z,W_1,\dots,W_l}(y,z) f_{Z|W_1,\dots,W_l}(z) \\ &= \frac{1}{2\pi\sigma_{Y|Z}\sigma_Z} \exp\left(\frac{-(y - \alpha - \beta z)^2}{2\sigma_{Y|Z}^2}\right) \exp\left(\frac{-(z - \sum_{i=0}^l (a_i + bc_i) W_i)^2}{2\sigma_Z^2}\right), \end{aligned}$$

where α , β and $\sigma_{Y|Z}^2$ are constants to be determined. The density of $Y | Z, W_1, \dots, W_l$ may be found directly by dividing the first expression for the joint density by the density of $Z | W_1, \dots, W_l$ (Bayes' theorem), or alternatively it may be deduced from the linear and quadratic terms in y in the exponential terms as follows. Equating the coefficients of y^2 yields

$$\frac{1}{2\sigma_{Y|Z}^2} = \frac{b^2}{2\sigma_{Z|Y}^2} + \frac{1}{2\sigma_Y^2}$$

from which it follows that the desired variance is

$$\sigma_{Y|Z}^2 = \frac{\sigma_{Z|Y}^2 \sigma_Y^2}{\sigma_{Z|Y}^2 + b^2 \sigma_Y^2}.$$

Equating the terms linear in y yields

$$\frac{\alpha + \beta z}{\sigma_{Y|Z}^2} = \frac{b(z - \sum_{i=0}^l a_i W_i)}{\sigma_{Z|Y}^2} + \frac{\sum_{i=0}^l c_i W_i}{\sigma_Y^2}$$

hence the conditional mean $\alpha + \beta Z$ is given by

$$\left(\frac{b(Z - \sum_{i=0}^l a_i W_i)}{\sigma_{Z|Y}^2} + \frac{\sum_{i=0}^l c_i W_i}{\sigma_Y^2} \right) \Big/ \left(\frac{\sigma_{Z|Y}^2 + b^2 \sigma_Y^2}{\sigma_{Z|Y}^2 \sigma_Y^2} \right) = \frac{\sum_{i=0}^l (c_i \sigma_{Z|Y}^2 - a_i b \sigma_Y^2) W_i + b \sigma_Y^2 Z}{\sigma_{Z|Y}^2 + b^2 \sigma_Y^2}$$

Thus,

$$Y | Z, W_1, \dots, W_l \sim N \left(\frac{\sum_{i=0}^l (c_i \sigma_{Z|Y}^2 - a_i b \sigma_Y^2) W_i + b \sigma_Y^2 Z}{\sigma_{Z|Y}^2 + b^2 \sigma_Y^2}, \frac{\sigma_Y^2 \sigma_{Z|Y}^2}{\sigma_{Z|Y}^2 + b^2 \sigma_Y^2} \right),$$

Case 2: $\sigma_Y > 0$ and $\sigma_{Z|Y}^2 = 0$.

We may deduce that

$$Y | Z, W_1, \dots, W_l \sim N \left(\frac{Z - \sum_{i=0}^l a_i W_i}{b}, 0 \right)$$

either by considering the limit $\sigma_{Z|Y}^2 \rightarrow 0$ in Case 1, or by noting that if $\sigma_{Z|Y}^2 = 0$, then

$$Z | Y, W_1, \dots, W_l \sim N(a_0 + a_1 W_1 + \dots + a_l W_l + bY, \sigma_{Z|Y}^2)$$

is equivalent to

$$Z = a_0 + a_1W_1 + \cdots + a_lW_l + bY$$

which is equivalent to the constraint

$$Y = \frac{Z - a_0 - a_1W_1 - \cdots - a_lW_l}{b}.$$

Case 3: $\sigma_Y = 0$ and $\sigma_{Z|Y}^2 \geq 0$.

We may obtain

$$Y | Z, W_1, \dots, W_l \sim N\left(\frac{Z - \sum_{i=0}^l a_i W_i}{b}, 0\right)$$

either as the limit $\sigma_Y^2 \rightarrow 0$ of Case 1, or by noting that the deterministic constraint implied by

$$Y | W_1, \dots, W_l \sim N\left(\sum_{i=0}^l c_i W_i, 0\right) \equiv Y = \sum_{i=0}^l c_i W_i$$

will be unaffected by further conditioning on Z .

References

- A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence Journal*, 2001.
- R. G. Cowell. Decision networks: a new formulation for multistage decision problems. *Research Report 132*, Department of Statistical Science, University College London, London, United Kingdom, 1994.
- R. G. Cowell, S. L. Lauritzen, and J. Mortera. Identification and separation of DNA mixtures using peak area information. Cass Statistical Science Research Report 25, City University London, 2004.
- R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999.
- A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.
- U. Kjærulff. Graph triangulation — algorithms giving small total state space. *Technical Report R 90-09*, Aalborg university, Denmark, 1990.
- P. Larrañaga, C. M. H. Kuijpers, M. Poza, and R. H. Murga. Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, pages 19–34, 1997.
- S. L. Lauritzen. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108, 1992.
- S. L. Lauritzen. *Graphical Models*. Oxford, United Kingdom, 1996.

- S. L. Lauritzen and N. Wermuth. Mixed interaction models. Technical Report R 84-8, Institute for Electronic Systems, Aalborg University, 1984.
- S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.
- S. L. Lauritzen and F. Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11:191–203, 2001.
- S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, Series B*, 50:157–224, 1988.
- H.-G. Leimer. Triangulated graphs with marked vertices. *Annals of Discrete Mathematics*, 41: 311–324, 1989.
- A. L. Madsen and F. V. Jensen. Lazy propagation in junction trees. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, California, 1998. Morgan Kaufmann, San Francisco, California.
- K. G. Olesen and A. Madsen. Maximal prime subgraph decomposition of Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 32(21–31), 2002.
- J. Pearl. *Probabilistic Inference in Intelligent Systems*. Morgan Kaufmann, San Mateo, California, San Mateo, California, 1988.
- C. Raphael. Bayesian networks with degenerate Gaussian distributions. *Methodology and Computing in Applied Probability*, 5(2):235–263, 2003.
- R. D. Shachter and C. Kenley. Gaussian influence diagrams. *Management Science*, 35:527–550, 1989.
- R. D. Shachter, S. K. Andersen, and K. L. Poh. Directed reduction algorithms and decomposable graphs. In *In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, July 27-29, Cambridge, MA*, pages 237–244, New York, NY, 1990. Elsevier Science Publishing Company, Inc.
- R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. 13:566–579, 1984.
- M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2:77–79, 1981.

A Bayesian Model for Supervised Clustering with the Dirichlet Process Prior

Hal Daumé III

Daniel Marcu

*Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292, USA*

HDAUME@ISI.EDU

MARCU@ISI.EDU

Editor: William Cohen

Abstract

We develop a Bayesian framework for tackling the supervised clustering problem, the generic problem encountered in tasks such as reference matching, coreference resolution, identity uncertainty and record linkage. Our clustering model is based on the Dirichlet process prior, which enables us to define distributions over the countably infinite sets that naturally arise in this problem. We add *supervision* to our model by positing the existence of a set of unobserved random variables (we call these “reference types”) that are generic across all clusters. Inference in our framework, which requires integrating over infinitely many parameters, is solved using Markov chain Monte Carlo techniques. We present algorithms for both conjugate and non-conjugate priors. We present a simple—but general—parameterization of our model based on a Gaussian assumption. We evaluate this model on one artificial task and three real-world tasks, comparing it against both unsupervised and state-of-the-art supervised algorithms. Our results show that our model is able to outperform other models across a variety of tasks and performance metrics.

Keywords: supervised clustering, record linkage, citation matching, coreference, Dirichlet process, non-parametric Bayesian

1. Introduction

Supervised clustering is the general characterization of a problem that occurs frequently in strikingly different communities. Like standard clustering, the problem involves breaking a finite set $X \subseteq \mathcal{X}$ into a K -way partition B_1, \dots, B_K (with K unknown). The distinction between supervised clustering and standard clustering is that in the supervised form we are given training examples. These training examples enable a learning algorithm to determine what aspects of X are relevant to creating an appropriate clustering. The N training examples $(X^{(n)}, \{B_k\}_{k=1 \dots K}^{(n)})$ are subsets of \mathcal{X} paired with their correct partitioning. In the end, the supervised clustering task is a prediction problem: a new $X^{(n+1)} \subseteq \mathcal{X}$ is presented and a system must produce a partition of it.

The supervised clustering problem goes under many names, depending on the goals of the interested community. In the relational learning community, it is typically referred to as *identity uncertainty* and the primary task is to augment a reasoning system so that it does not implicitly (or even explicitly) assume that there is a one-to-one correspondence between elements in an knowledge base and entities in the real world (Cohen and Richman, 2002; Pasula et al., 2003). In the

database community, the task arises in the context of merging databases with overlapping fields, and is known as *record linkage* (Monge and Elkan, 1997; Doan et al., 2004). In information extraction, particularly in the context of extracting citations from scholarly publications, the task is to identify which citations are to the same publication. Here, the task is known as *reference matching* (McCallum et al., 2000). In natural language processing, the problem arises in the context of *coreference resolution*, wherein one wishes to identify which entities mentioned in a document are the same person (or organization) in real life (Soon et al., 2001; Ng and Cardie, 2002; McCallum and Wellner, 2004). In the machine learning community, it has additionally been referred to as *learning under equivalence constraints* (Bar-Hillel and Weinshall, 2003) and *learning from cluster examples* (Kamishima and Motoyoshi, 2003).

In this paper, we propose a generative model for solving the supervised clustering problem. Our model takes advantage of the *Dirichlet process prior*, which is a non-parametric Bayesian prior over discrete distributions. This prior plays two crucial roles: first, it allows us to estimate the number of clusters K in a principled manner; second, it allows us to control the complexity of the solutions that are learned. We present inference methods for our model based on Markov chain Monte Carlo methods. We compare our model against other methods on large, real-world data sets, where we show that it is able to outperform most other systems according to several metrics of performance.

The remainder of this paper is structured as follows. In Section 2, we describe prior efforts to tackle the supervised clustering problem. In Section 3, we develop our framework for this problem, starting from very basic assumptions about the task. We follow this discussion with a general scheme for inference in this framework (Section 4). Next, in Section 5, we present three generic parameterizations of our framework and describe the appropriate adaptation of the inference scheme to these parameterizations. We then discuss performance metrics for the supervised clustering problem in Section 6 and present experimental results of our models’ performance on artificial and real-world problems in Section 7. We conclude in Section 8 with a discussion of the advantages and disadvantages of our model, our generic parameterization, and our learning techniques.

2. Prior Work

The most common technique for solving supervised clustering is by mapping it to binary classification. For a given input set, a binary classifier is trained on all pairs of inputs, eliciting a positive output if the two elements belong in the same cluster and a negative output otherwise. When applied to test data, however, such a classifier will not necessarily produce a valid equivalence relation (i.e., it might say $x = y$ and $y = z$ but $x \neq z$); to solve this problem, the outputs of the binary classifier are fed into a clustering algorithm. Among others, Cohen and Richman (2002) present an agglomerative clustering algorithm in the task of record linkage; Bar-Hillel and Weinshall (2003) present a similar, but more complex algorithm that is provably optimal whenever the binary classifier is sufficiently good.¹

The binary classification plus clustering approach is attractive primarily because both of these problems have individually received much attention; thus, good algorithms are known to solve them. The primary disadvantages of these approaches are the largely ad-hoc connection between the clas-

1. Unfortunately, the “sufficiently good requirement” of Bar-Hillel and Weinshall (2003) is often unattainable: it states that the classifier must achieve an error rate of at most $R^2/6$, where R is the ratio of the size of the smallest class to the total number of points. In many real world problems, the size of the smallest class is 1, and the number of points is quite large, meaning that only a perfect classifier will achieve the required accuracy.

sifier and the clustering algorithm, the necessity of training over $O(n^2)$ data points, and the potential difficulty of performing unbiased cross-validation to estimate hyperparameters. The first issue, the ad-hoc connection, makes it difficult to make state precise statements about performance. The second can cause computational problems for expensive classifiers (such as SVMs) and invalidates the i.i.d. assumption that is necessary for many generalization bounds.² The final issue, regarding cross-validation, has to do with the fact that the classification plus clustering approach is based on pipelining two independent systems (see Section 7.1 for how the cross-validation is done in our comparative model).

In addition to the classification plus clustering approach, there have been several attempts to solve the supervised clustering problem directly. Some researchers have posed the problem in the framework of learning a distance metric, for which, eg., convex optimization methods can be employed (Bar-Hillel et al., 2003; Xing et al., 2003; Basu et al., 2003). Using a learned distance metric, one is able to use a standard clustering algorithm for doing the final predictions. These methods effectively solve all of the problems associated with the classification plus clustering approach. The only drawback to these approaches is that they assume Euclidean data and learn a Mahalanobis distance metric. It is often unclear how to extend this assumption to a more general space or a more general notion of similarity.

Two other recent techniques have been proposed for directly solving the supervised clustering problem, and are not phrased in terms of learning a Mahalanobis distance. The first, due to McCallum and Wellner (2004), is based on conditional random fields. In this model, a fully connected graph is created, where nodes are elements in a data set. Feature functions are defined over the edges (corresponding to pairs of input elements), and weights are learned to maximize the conditional likelihood of the data. In order to ensure that the model never predicts intransitive solutions, clique potentials of $-\infty$ are inserted for any solution that is intransitive. Exact inference in this model is intractable (as in most supervised clustering models), and they employ a simple perceptron-style update scheme, which they show to be quite effective on this task. The perceptron requires that the most likely clustering be found for a given set of weights, which is NP-complete by reduction to graph partitioning; McCallum and Wellner (2004) employ a standard approximation algorithm for performing this operation. This technique appears promising, largely because it can incorporate arbitrary feature functions. The only potential drawback seems to be that two approximations are used: the perceptron approximation to the CRF likelihood³ and an approximate graph partitioning algorithm for performing the clustering.

The other direct solution to the supervised clustering problem, due to Finley and Joachims (2005), is based on the SVMs for Interdependent and Structured Outputs technique (Tsochantaridis et al., 2004). In this model, a particular clustering method, *correlation clustering*, is held fixed, and weights are optimized to minimize the regularized empirical loss of the training data with respect to this clustering function. The choice of correlation clustering is not accidental: it decomposes over pairs. The advantage of this model over the model of McCallum and Wellner (2004) is primarily due to the fact that the SVM model can optimize more complex (and appropriate) loss functions than can the CRF approach. However, like the CRF approach, the SVMISO approach must resort to approximation methods for finding solutions during learning.

2. For instance, the pairs (x_1, x_2) and (x_3, x_4) can be seen as being drawn i.i.d. from a joint pair distribution, but the pairs (x_1, x_2) , (x_2, x_3) cannot possibly be i.i.d.

3. It could be argued that the perceptron “approximation” is actually superior to the CRF, since it optimizes something closer to “accuracy” than the log-loss optimized by the CRF.

In comparison to other models that have been proposed, ours most closely resembles the (non-Bayesian) generative model proposed by Pasula et al. (2003). This model formulates the identity uncertainty/citation matching problem in a generative framework, based on a complex generative model under which inference is intractable. They resort to an Markov chain Monte Carlo inference scheme for identifying clusters, where a uniform prior is placed on the number of clusters. Their framework learns the model parameters through an MCMC sampling procedure, though no learning is done with respect to the prior on the number of clusters. The work we present in this paper can be seen as a method for extending their approach in two ways: first, we directly model the number of output clusters; second, we provide an intuitive, effective procedure for accounting for the multiple aspects of similarity between different instances. As we discuss in Section 8, the hybridization of their model and the one we propose could lead to a more effective system than either alone. (Indeed, between the time of submission of this paper and its final acceptance, Carbonetto et al. (2005) have presented an extension to the Pasula et al. (2003) model that solves the first problem: estimating the number of clusters in the citation matching domain. Like us, they employ a Dirichlet process model to solve this problem. The fact that this model has now been proposed twice, independently, is not surprising: citation matching is a well-known problem that suffers from the need to estimate the number of clusters in a data set, and the Dirichlet process excels at precisely this task.)

3. Supervised Clustering Model

In this section, we describe our model for the supervised clustering problem. To facilitate discussion, we take our terminology and notation from the reference matching task. The canonical example of this task is the CiteSeer/ResearchIndex database. Specifically, we assume that we are given a list of references appearing in the bibliographies of scholarly publications and that we need to identify which references correspond to the same publication. This task is difficult: according to CiteSeer, there are currently over 100 different books on *Artificial Intelligence* by Russell and Norvig, according to Pasula et al. (2003). We refer to the set \mathcal{X} as the set of *references* and a correct cluster of references as a *publication*. In our problem, the observed data is a set of references paired with partial equivalence classes over those references (partial publications). For instance, we might know that $r_1, r_2, r_3 \in \mathcal{X}$ belong to the same equivalence class (are the same publication), but we might not have any information about the equivalence class of r_4 . In this case, we identify r_1, r_2, r_3 as training data and r_4 as test data.

In general, we have a countable set of references \mathcal{X} and some information about the structure of equivalence classes on this set and seek to extend the observed equivalence classes to all of \mathcal{X} . In complete generality, this would be impossible, due to the infinite nature of \mathcal{X} and the corresponding equivalence classes. However, in the *prediction* case, our job is simply to make predictions about the structure of a *finite* subset of \mathcal{X} , which we have previously denoted $X^{(n+1)}$. Thus, while our inference procedure attempts to uncover the structure of an infinite structure, calculations are possible because at any given time, we only deal with a finite portion of this set. This is not unlike the situation one encounters in Gaussian processes, wherein a distribution is placed over a function space, but computations are tractable because observations are always finite.

3.1 Generative Story

The model we describe is a generative one. Our modeling assumption is that a reference is generated according to the cross-product of two attributes. The first attribute specifies which publication this

reference belongs to. The second attribute specifies the manner in which this reference is created, which we call the “reference type.” A reference type encompasses the notion that under different circumstances, references to the same publication are realized differently.

In the terminology of reference matching, in the context of a short workshop paper (for instance), author first names might be abbreviated as initials, page numbers might be left off and conferences and journals might be referred to by abbreviations. On the contrary, in a reference appearing in a journal, page numbers are included, as are full conference/journal names and author names. In the context of coreference resolution, one reference type might be for generating proper names (“Bill Clinton”), one for nominal constructions (“the President”) and one for pronouns (“he”). Of course, the form and number of the reference types is unknown.

The generative process for a data set proceeds as follows:

1. Select a distribution G_0^p over publications that will be referred to in this data set. G_0^p should assign positive probability to only a finite set of all possible publications.
2. Select a distribution G_0^t over reference types that will be used in this data set; again, G_0^t should be finite.
3. For each reference r_n appearing in the data set:
 - (a) Select the corresponding publication $p_n \sim G_0^p$.
 - (b) Select the corresponding reference type $t_n \sim G_0^t$.
 - (c) Generate r_n by a problem-specific distribution parameterized by the publication and reference type: $r_n \sim F(p_n, t_n)$.

The difficulty with this model is knowing how to parameterize the selection of the distributions G_0^p and G_0^t in steps 1 and 2. It turns out that a Dirichlet process is an excellent tool for solving this problem. The Dirichlet process (DP), which is a *distribution over distributions*, can be most easily understood via a generalized Pòlya urn scheme, where one draws colored balls from an urn with replacement. The difference is that when a black ball is drawn, one replaces it together with a ball of a new color. In this way, the number of “classes” (ball colors) is unlimited, but defines a discrete distribution (with probability one). See Appendix A for a brief review of the properties of the DP that are relevant to our model.

Our model is seen as an extension of the standard naïve-Bayes multiclass classification model (in the Bayesian framework), but where we allow the number of classes to grow unboundedly. Just as a multiclass classification model can be seen as a finite mixture model where the mixture components correspond to the finite classes, the supervised clustering model can be seen as an *infinite* mixture model. In the case of the standard multiclass setup, one treats the class y as a random variable drawn from a multinomial distribution $\mathcal{Mult}(\pi)$, where π is again a random variable with prior distribution $\mathcal{Dir}(\alpha)$ for the standard Dirichlet distribution. In our model, we essentially remove the requirement that there is a known finite number of classes and allow this to grow unboundedly. In order to account for the resulting non-identifiability of the classes, we introduce the notion of reference types to capture the relationships between elements from the same class.

Whenever one chooses a model for a problem, it is appropriate to ascertain whether the chosen model is able to adequately capture the required aspects of a data set. In the case of our choice of the Dirichlet process as a prior over publications, one such issue is that of the expected number

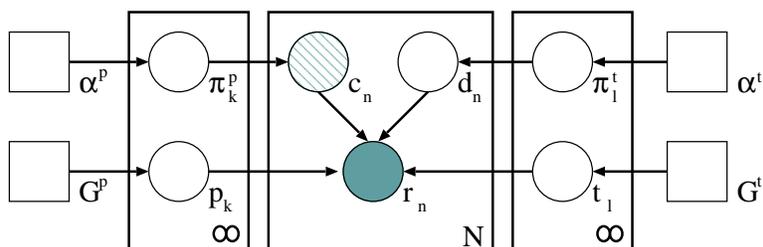


Figure 1: Graphical model for our generic supervised clustering model.

of publications per citation. We have performed such experiments and verified that on a variety of problems (reference matching, identity uncertainty and coreference resolution), the Dirichlet process is appropriate with respect to this measure (see Section 7.3 and Figure 3 for discussion).

3.2 Hierarchical Model

The model we propose is structured as follows:

$$\begin{aligned}
 \pi^p \mid \alpha^p &\sim \text{Dir}(\alpha^p/K, \dots, \alpha^p/K) & \pi^t \mid \alpha^t &\sim \text{Dir}(\alpha^t/L, \dots, \alpha^t/L) \\
 c_n \mid \pi^p &\sim \text{Disc}(\pi_1^p, \dots, \pi_K^p) & d_n \mid \pi^t &\sim \text{Disc}(\pi_1^t, \dots, \pi_L^t) \\
 p_k \mid G_0^p &\sim G_0^p & t_k \mid G_0^t &\sim G_0^t \\
 r_n \mid c_n, d_n, p, t &\sim F(p_{c_n}, t_{d_n})
 \end{aligned} \tag{1}$$

The corresponding graphical model is depicted in Figure 1. In this figure, we depict the α and G parameters as being fixed (indicated by the square boxes). The α s give rise to multinomial random variables π , which in turn determine indicator variables c_n (specifying the publication to which r_n belongs) and d_n (specifying the reference type used by reference r_n). The base density G^p generates publications p_k (according to a problem-specific distribution), while the base density G^t generates reference types t_l (again according to a problem-specific distribution). Finally, the observed reference r_n is generated according to publication p_{c_n} and reference type t_{d_n} with problem-specific distribution F . The r_n random variable (the reference itself) is shaded to indicate that it is always observed, and the c_n random variable (the indicator as to which publication is used for reference r_n) is partially shaded to indicate that it is sometimes observed (in the training data) and sometimes not (in the test data).

As indicated by the counts on the plates for the (π^p, p) and (π^t, t) variables, we take the limit as $K \rightarrow \infty$ and $L \rightarrow \infty$ (where K is the number of publications and L is the number of reference types). This limit corresponds to a choice of a Dirichlet process prior on the ps and ts (Neal, 1998).

4. Inference Scheme

Inference in infinite models differs from inference in finite models, primarily because we cannot store all possible values for infinite plates. However, as noted earlier, we only encounter a finite amount of data, so at any time only a finite number of these infinite parameters will be active—i.e., only a finite number of them will affect the distribution of the observed data. We will suggest and implement inference schemes based on Markov chain Monte Carlo (MCMC) techniques, which are the most frequently used methods for inference in DP models (Antoniak, 1974; Escobar, 1994;

Neal, 1998; MacEachern and Müller, 1998; Ishwaran and James, 2001; Beal et al., 2002; Xing et al., 2004). Recently, Blei and Jordan (2005) have presented a variational approach to Dirichlet process models, and Minka and Ghahramani (2004) have presented an inference procedure for DP models based on expectation propagation. Unfortunately, these methods do not work when the prior distributions G_0 are not conjugate to the data distribution F and they are thus not of use to us.

The MCMC-based Bayesian solution to the supervised clustering problem (or, indeed, any problem) is to write down the expression corresponding to the posterior distribution of the c_n s for the test data and draw samples from that posterior. Writing data points 1 through N as the training data and points $N + 1$ through $N + M$ as the test data, we obtain the following expression for this posterior (the actual distributions are from Equation (1)):

$$p(c_{N+1:N+M} | r_{1:N+M}, c_{1:N}) \propto \int d\pi^p p(\pi^p | \alpha^p) \int d\pi^t p(\pi^t | \alpha^t) \int dp p(p | G_0^p) \int dt p(t | G_0^t) \sum_{d_{1:N+M}} \prod_{n=1}^{N+M} p(c_n | \pi^p) p(d_n | \pi^t) p(r_n | p_{c_n}, t_{d_n}).$$

We now describe how we can do this sampling. Most of the information in this section is taken from Neal (1998), in which a vast amount of additional information is provided. The interested reader is directed there for additional motivation and different algorithms. The algorithms we use in this paper are either exact replicas, or slight deviations from Algorithms 2 and 8 of Neal’s.

4.1 Updates for Conjugate Priors

The simplest case arises when a conjugate prior is used. In the terminology of the Dirichlet process, this means that the data sampling distribution F is conjugate to the base density G_0 of the Dirichlet process. To perform inference with conjugate priors, we need to be able to compute the marginal distribution of a single observation and need to be able to draw samples from the posterior of the base distributions. In each iteration of sampling, we first resample each active publication p_c and reference type t_d according to their posterior densities (in the case of conjugate priors, this is possible). Then, for each test reference, we resample its publication and for all references, we resample the corresponding reference type. The algorithm is shown in Figure 2. We actually have two options when sampling the c_n s, depending on whether publications are allowed to be shared across the training and testing data. If a training reference may refer to the same publication as a testing reference (as is natural in the context of reference matching), then the sum in Equation (2) is over all data; on the other hand, if they are not allowed to co-refer (as is natural in, for example, single-document coreference resolution), then the sum is only over the test data.

4.2 Updates for Non-Conjugate Priors

The case of non-conjugate priors is a bit more complex, since in this case, in general, one is not able to analytically compute the data marginals, nor is one able to directly sample from the relevant posterior distributions. A naïve solution would be to set up separate Markov chains to draw samples from the appropriate distributions so that we *could* calculate these. Unfortunately, since these values need to be computed for each loop of the “outer” Markov chain, such an approach is impractical. The alternative—given as Algorithm 8 by Neal (1998)—is essentially to sample just a few of these

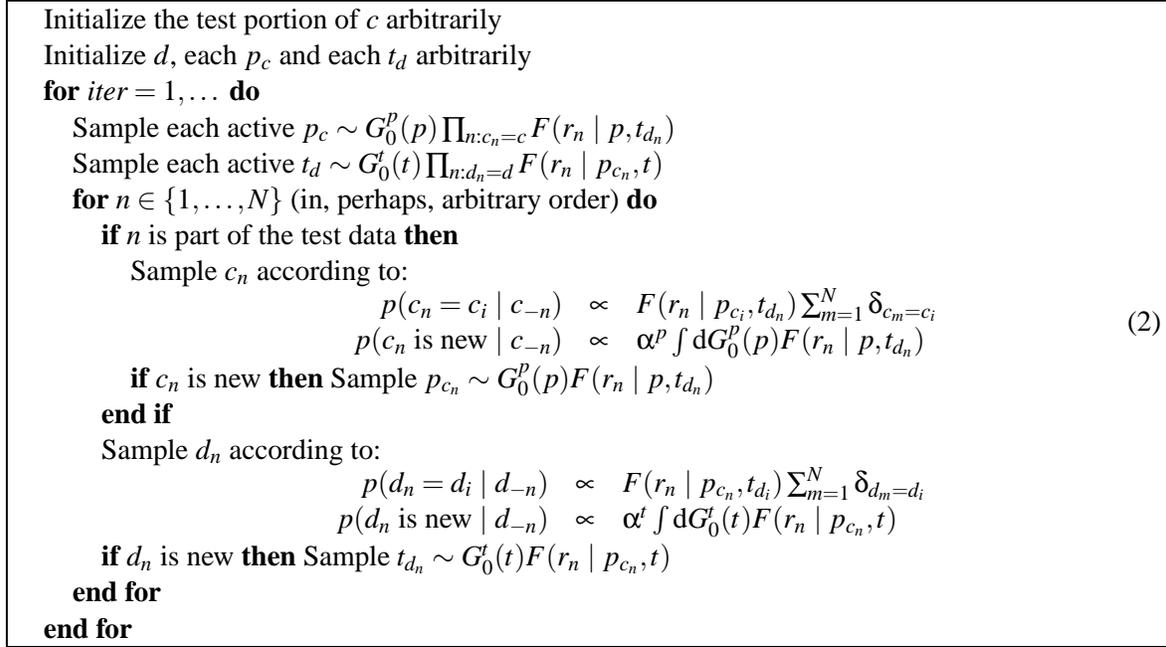


Figure 2: The inference algorithm for the supervised clustering model with conjugate priors.

needed values in a way that does not affect the detailed balance condition that guarantees that the *outer* Markov chain converges to the correct stationary distribution.

The overall structure of the sampling algorithm remains identical in the case of non-conjugate priors; however, the sampling for the indicator variables c_n and d_n changes slightly, and so does the sampling of the p and t variables. For instance, in the conjugate case, d_n is sampled according to the marginal distribution $\int dG_0^t(t) F(r_n | p_{c_n}, t)$, which is analytically unavailable when G_0^t is not conjugate to F (with respect to the second variable). In the case of non-conjugacy, we approximate this integral by drawing \tilde{M} samples independently from G_0^t . In general, as $\tilde{M} \rightarrow \infty$, this is exactly like computing the integral with an independence sampler; however, for \tilde{M} finite, we still get convergence of the overall Markov chain. \tilde{M} is set by the experimenter by choosing the number of samples M that is drawn and then setting \tilde{M} to be M whenever the old value of d_n was not unique, and to $M + 1$ whenever it was unique. If the chosen value corresponds to one of the newly sampled t s, then we set t_d to be that sampled value. The corresponding sampling for the c variables is identical. This is the technique suggested by Neal (1998) in his Algorithm 8. In all experiments, we use $M = 8$.

The second complication is when we cannot sample from the data posteriors, which means that resampling p and t is difficult. This is partially assuaged by the fact that in sampling for c_n and d_n we are given an explicit new value of p or t to use. However, at the beginning of each iteration of the chain, we must resample p according to its posterior distribution (and similarly for t). The most general approach to solving this problem—and the approach we employ here—is to run a short independence sampler for p by drawing a set of values p from G_0^p and then choosing one of those according to its posterior. However, depending on the actual distributions chosen, there might be more appropriate methods for doing this sampling that still leaves the overall chain invariant.

4.3 Resampling the Dirichlet Process Precision

We often wish to leave the values of α^p and α^t (the scaling/precision hyperparameters for the two Dirichlet processes) as random variables, and estimate them according to the data distribution. West (1992) gives a method for drawing samples for the precision parameter given the number of references N and the number of publications K (or, for α^t , the number of reference types); in his analysis, it is natural to place a gamma prior on α . In most cases, his analysis can be applied directly; however, in the case of coreference resolution, the problem is a bit more complicated because we have *multiple* observations pairs (N, K) for each “training document.” In Appendix B, we briefly extend this analysis to the case where there are multiple observations.

5. Model Parameterization

One of the simplest model parameterizations occurs when the data points r_n are vectors in the Euclidean space \mathbb{R}^F for some dimensionality F , and when each dimension is a measure of distance (i.e., $|r_{nf} - r_{mf}|$ is small whenever r_n and r_m are similar along dimension f). In this case, it may be a reasonable assumption that the r_n s are distributed normally around some unknown mean vector, and with some unknown covariance. While the assumption of normalcy is probably not accurate, it turns out that it fares rather well experimentally (see Section 7). Moreover, as discussed at the end of this paper, it is possible to substitute in other models for F as deemed appropriate by a specific problem.

If we believe the r_n s are distributed normally (i.e., F is a Normal distribution), it is natural to treat the p_k variables as means and the t_l variables as precisions (inverse variance-covariances matrices). For efficiency’s sake, we further assume that t_l is *diagonal*, so that all covariance terms are zero. In this model, one can think of a precision t_{lf} as the “weight” along dimension f , so that high weights mean that this dimension is important and low weights mean that this dimension is not relevant.

By making F an isotropic Normal distribution, the natural conjugate priors are to make G_0^p another Normal distribution and to make G_0^t a product of inverse-gamma distributions (one inverse-gamma distribution per dimension f).⁴ As we typically center and spherize the training data, it is natural to parameterize G_0^p with a mean of 0 and a covariance matrix of $\sigma \mathbf{I}$ for some $\sigma \approx 1$. Similarly, we may parameterize G_0^t with identical scale and shape parameters all approximately 1. (Note that we could also *learn* these hyperparameters during inference by including them in the sampling, though we do not explore this option.)

Experiments with the model just described have demonstrated that while it is adept at finding points in the same cluster, it is not as able to separate out points in different clusters (it has low precision, in the precision/recall sense). This occurs because the Gaussian precisions are learned solely for the purpose of accounting for the distribution of classes by themselves, but with no regard to the relation between classes. We explore two modeling extensions to attempt to alleviate this problem and give the model a better ability to separate classes; in the first, we maintain conjugacy (and hence efficiency in implementation), but in the second we give up conjugacy for a more appropriate model.

4. If we had not assumed that t was diagonal, then the natural choice for G_0^t would be an inverse-Wishart distribution.

5.1 Separation by Modifying G_0^p

Our first method for adding separation power between to the model is to condition the parameters of G_0^p on p and c : in other words, the shape and scale parameters of the prior on the precisions is affected by the relative positions of the means of the data. In the original model, we assumed that $t_f \sim \text{Gam}(1, 1)$ is a gamma random variable with mean 1 and variance 1. Here, we wish to change this distribution so that the mean is large enough to keep the data separated along this dimension, and the variance is small whenever many points tell us that this dimension is important. To accomplish this we use instead a $\text{Gam}(a, b)$ prior, where ab is half the mean variance along dimension f and ab^2 is the variance of the variance along dimension f . The values for a and b must be resampled at each iteration of the algorithm.

5.2 Separation by Conditioning

Our second approach to adding more separation power to the model is to condition the choice of the precisions (reference types) t on the means (publications) p . In terms of our generative story, this means that first we choose a publication then, based on the publication, choose a reference type. Since we wish to ascribe no meaning to the actual location of the means p_k , we compute this probability based only on their relative distances (along each dimension), and also under a naïve Bayes assumption:

$$\begin{aligned}
 p(t \mid p, c, d) &\stackrel{[1]}{\approx} \prod_{i=1}^{|d|} p(t_i \mid p, c, d) \\
 &\stackrel{[2]}{=} \prod_{i=1}^{|d|} \frac{p(t_i \mid c, d) p(p \mid t_i, c, d)}{p(p \mid c, d)} \\
 &\stackrel{[3]}{\approx} \prod_{i=1}^{|d|} \frac{G_0^t(t_i)}{\prod_{j=1}^{|c|} G_0^p(p_j)} \prod_{j=1}^{|c|} p(p_j \mid t_i, p_{1:j-1}, c, d) \\
 &\stackrel{[4]}{=} \prod_{i=1}^{|d|} G_0^t(t_i) \prod_{j=1}^{|c|} \frac{p(p_j \mid t_i, c, d) p(p_{1:j-1} \mid t_i, p_j, c, d)}{p(p_{1:j-1} \mid t_i, c, d) G_0^p(p_j)} \\
 &\stackrel{[5]}{\approx} \prod_{i=1}^{|d|} G_0^t(t_i) \prod_{j=1}^{|c|} \frac{p(p_j \mid t_i, c, d) \prod_{k=1}^{j-1} p(p_k \mid t_i, p_j, c, d)}{G_0^p(p_j) \prod_{k=1}^{j-1} p(p_k \mid t_i, c, d)} \\
 &\stackrel{[6]}{=} \prod_{i=1}^{|d|} G_0^t(t_i) \prod_{j=1}^{|c|} G_0^p(p_j)^{2(j-1)-|c|} \prod_{k=1}^{j-1} p(p_j \mid p_k, t_i). \tag{3}
 \end{aligned}$$

In the first step of this derivation, we make a factorial assumption on the t vector. The second step simply applies Bayes' rule. The third step replaces the generic $p(\cdot)$ symbol for the t_i variables with the true distribution G_0^t , makes a similar factorial assumption on the p vector and replaces the corresponding $p(\cdot)$ with G_0^p . The fourth step applies Bayes' rule to the last term and moves the denominator from the first product into the second. The fifth step applies the same factorial assumption on $p_{1:j-1}$ as before. The last step replaces the generic $p(\cdot)$ symbol with G_0^p and performs some minor algebraic manipulation.

This final expression in Equation (3) depends only on the prior values for the sampled t s and p s, coupled with the probability of mean p_j given p_k under precision t_i . Unfortunately, under the assumptions made, the probability of a vector p is no longer independent of the ordering of the values of p . In all our experiments, we order the p according to the sizes of the classes: if $\text{count}(c_1) > \text{count}(c_2)$. We parameterize the distribution on the means $p(p_j | p_k, t_i)$ by treating the *distance* between p_j and p_k , measured by t_i as a random variable with an exponential distribution: $p(p_j | p_k, t_i) = \lambda \exp[-\lambda \|p_j - p_k\|_{t_i}^2]$. We set $\lambda = 1$, but, again, it could be learned concurrently by sampling.

Clearly, this prior distribution for t is no longer conjugate to the data sampling distribution F . Moreover, the p s and t s are no longer separated by the indicator variables, which makes the entire sampling story more complex. Indeed, the marginal distribution now depends on the types and, similarly, the types depend on the mentions. We thus use the non-conjugate updates described in Section 4.2. The simplest approach to performing inference with the non-conjugate priors would be, for each of the \tilde{M} samples for p , to draw from G_0^p and weight the sampled \tilde{p} s proportional to its unnormalized posterior probability, given by Equation (3). Similarly, a proposed sample \tilde{t} would be weighted according to its (unnormalized) posterior probability according to Equation (3).

6. Performance Metrics

Quite a few performance metrics have been proposed in the literature for comparing two clusterings of a given data set. Since these are, in general, less well known than the metrics used for classification (accuracy, ROC, etc.), we review them here, and attempt to point out the strengths and weaknesses of each metric. Of course, the evaluation criteria one uses should reflect one’s own personal views of what is important, but the metrics used here can be seen as surrogate measurements when such prior knowledge is unavailable. All of these metrics assume that we have a gold standard (correct) clustering G and a hypothesis clustering H and that the total number of data points is N .

6.1 Rand Index

The rand index (Rand, 1971) is computed by viewing the clustering problem as a binary classification problem. Letting N_{11} denote the number of pairs that are in the same cluster in both G and in H , and letting N_{00} denote the number of pairs that are in different clusters in both G and H , the rand index has value $\mathbf{RI}(G, H) = 2[N_{11} + N_{00}]/[N(N - 1)]$. Thus, the rand index computes the number of correct binary decisions ($N_{11} + N_{00}$) made by the system and normalizes by the total number of decisions made. The value of the rand index lies between 0 and 1, with 1 representing a perfect clustering.

The rand index is the most frequently reported metric in the clustering literature, though we believe that its value is often misleading. As we show in our results (Section 7), a very simple baseline system that places each element in its own cluster tends to achieve a very high rand index. This occurs due to the structure of the clusters in most real world data sets. In such data sets, the number of negative pairs (pairs that, in the gold standard, fall into different clusters) vastly outnumber the number of positive pairs; thus the rand index becomes dominated by the N_{00} factor, and the N_{11} factor tends to have very little impact on the final value. Moreover, the influence of large clusters on the rand index quadratically outnumbers the influence of small clusters on this value, so system performance on small clusters (which are typically the most difficult) becomes insignificant.

In this paper, we report the rand index for comparative purposes with earlier work, but strongly encourage readers not to take these numbers too seriously. We recommend other researchers in the supervised clustering field to report on other metrics of system performance than the rand index.

6.2 Precision, Recall, F-score

The second set of metrics we report are the precision/recall/F-score of the clustering. Extending the notation used for the rand index, we write N_{10} for the number of pairs that are in the same cluster in G , but in different clusters in H . Similarly, we write N_{01} for the number of pairs that are in different clusters in G but the same cluster in H . Precision is $\mathbf{P}(G, H) = N_{11}/[N_{11} + N_{01}]$, recall is $\mathbf{R}(G, H) = N_{11}/[N_{11} + N_{10}]$ and F-score is $\mathbf{F}(G, H) = (\mathbf{P}(G, H)^{-1} + \mathbf{R}(G, H)^{-1})^{-1}$. Again, each of these values falls between 0 and 1 with 1 being optimal. While precision, recall and F-score are still computed based on binary decisions, they do not suffer as strongly from the weaknesses of the rand index. However, they still place quadratically as much importance on large clusters.

6.3 Cluster Edit Distance and Normalized Edit Score

Pantel (2003) proposes a metric called the *cluster edit distance*, which computes the number of “create,” “move,” and “merge” operations required to transform the hypothesis clustering into the gold standard. Since no “split” operation is allowed, the cluster edit distance can be computed easily and efficiently. However, the lack of a split operation (which is absent precisely so that the computation of the metric is efficient) means that the cluster edit distance favors algorithms that tend to make too many clusters, rather than too few clusters. This is because if an algorithm splits an m element cluster in half, it requires only one merge operation to fix this; however, if, instead, two $m/2$ -sized clusters are mistakenly merged by an algorithm, $m/2$ operations are required to fix this error. The cluster edit distance has a minimum at 0 for the perfect clustering and a maximum of N . Also note that the cluster edit distance is not symmetric: in general, it does not hold that $\mathbf{CED}(G, H) = \mathbf{CED}(H, G)$ (again, precisely because splits are disallowed).

We propose a variant of the cluster edit distance that we call the *normalized edit score*. This value is computed as $\mathbf{NES}(G, H) = 1 - [\mathbf{CED}(G, H) + \mathbf{CED}(H, G)]/[2N]$ and is clearly symmetric and no longer favors fine clusterings over coarse clusterings. Additionally, it takes values from 0 to 1, with 1 being a perfect clustering. While the normalized edit score no longer can be interpreted in terms of the number of operations required to transform the hypothesis clustering into the correct clustering, we believe that these additional properties are sufficiently important to make it preferable to the cluster edit distance metric.

6.4 Variation of Information

The final metric we report in this paper is the variation of information (**VI**), introduced by Meila (2003). The **VI** metric essentially looks at how much entropy there is about G knowing H , and how much entropy there is about H knowing G . It is computed as $\mathbf{VI}(G, H) = H(G) + H(H) - 2I(G, H)$. Here, $H(\cdot)$ is the entropy of a clustering, computed by looking at the probability that any given point is in any particular cluster. $I(G, H)$ is the mutual information between G and H , computed by looking at the probability that two points are in the same cluster, according to G and H . It has a minimum at 0, only when the two clusterings match, and is bounded above by $\log N$. It has several other desirable properties, including the fact that it is a metric. Though frowned upon by Meila

(2003), we also report the *normalized variation of information*, computed simply as $\mathbf{NVI}(G, H) = 1 - \mathbf{VI}(G, H) / \log N$. This value is again bounded between 0 and 1, where 1 represents a correct clustering.

7. Experimental Results

In this section, we present experimental results on both artificial and real-world data sets, comparing our model against other supervised clustering algorithms as well as other standard clustering algorithms. We first discuss the baselines and systems we compare against, and then describe the data sets we use for comparison. Some data sets support additional, problem-specific baselines against which we also compare.

7.1 Systems Compared

The first baseline we compare against, COARSE, simply places all elements in the same, single cluster. The second baseline, FINE, places each element in its own cluster. These are straw-man baselines that are used only to provide a better sense of the performance metrics.

The next systems we compare against are pure clustering systems that do not perform any learning. In particular, we compare against K-MEANS, where the number of clusters, k , is chosen according to an oracle (this is thus an *upper bound* on how well the k-means algorithm can perform in real life). We additionally compare against a version of our model that does not use any of the training data. To do so, we initialize $\alpha^p = 1$ and use a single reference type, the identity matrix. This system is denoted CDP (for “Clustering with the Dirichlet Process”) in subsequent sections.

The final class of systems against which we compare are true learning systems. The first is based on the standard technique of building a binary classifier and applying a clustering method to it. We use an SVM as the classifier, with an RBF kernel. The kernel parameter γ and the regularization parameter C are tuned using golden section search under 10-fold cross validation. After the SVM has been optimized, we use an agglomerative clustering algorithm to create clusters according to either minimum, maximum or average link, with a threshold to stop merging. The link type (min, max or avg) and the threshold is tuned through another series of 10-fold cross validation on the training data. This is essentially the method advocated by Cohen and Richman (2002), with the slight complication that we consider all link types, while they use average link exclusively. This system is denoted BINARY in subsequent sections.

The second learning system is the model of distance metric learning presented by Xing et al. (2003). This model learns a distance metric in the form of a positive semi-definite matrix \mathbf{A} and computes the distance between vectors x and y as $[(x - y)^\top \mathbf{A} (x - y)]^{1/2}$. The matrix is learned so as to minimize the distances between elements in the same cluster (in the training data) and maximize the distance between elements in different clusters. Once this distance metric is learned, Xing et al. (2003) apply standard k-means clustering to the test data. There is a weighting term C that controls the trade-off between keeping similar points close and dissimilar points separate; we found that the performance of the resulting system was highly sensitive to this parameter. In the results we present, we ran four configurations, one with $C = 0$, one with $C = 1$, one with $C = |s|/|d|$ (where s is the set of similar points and d is the set of dissimilar points), and one with $C = (|s|/|d|)^2$. We evaluated all four and chose the one that performed best on the test data according to F-score (using an “oracle”). In all cases, either $C = 0$ or $C = (|s|/|d|)^2$ performed best. We denote this model XING-K in the following.

Lastly, we present results produced by the system described in this paper. We report scores on several variants of our “Supervised Clustering with the Dirichlet Process” model: SCDP-1 is the result of the system run using the conjugate inference methods; SCDP-2 is the model presented in Section 5.1 that is aimed at achieving better class separation by modifying G_0^p ; finally, SCDP-3 is the model presented in Section 5.2 that separates classes through conditioning. For all problems, we will report the number of iterations of the sampling algorithm run, and the time taken for sampling. In all cases, we ran the algorithms for what we *a priori* assumed would be “long enough” and did not employ any technique to determine if we could stop early.

7.2 Data Sets

We evaluate these models on four data sets, the first of which is semi-artificial, and the last three of which are real-world data sets from three different domains. The four data sets we experiment on are: the USPS digits database (1987), a collection of annotated data for identity uncertainty from Doan et al. (2004), proper noun coreference data from NIST and reference matching data from McCallum et al. (2000). In the digits data set, the data points live in a high-dimensional Euclidean space and thus one can directly apply all of the models discussed above. The last three data sets all involve textual data for which an obvious embedding in Euclidean space is not available. There are three obvious approaches to dealing with such data. The first is to use a Euclidean embedding technique, such as multidimensional scaling, kernel PCA or LLE, thus giving us data in Euclidean space to deal with. The second is to modify the Gaussian assumption in our model to a more appropriate, problem-specific distribution. The third, which is the alternative we explore here, is to notice that in all the computations required in our model, in k-means clustering, and in the distance metric learning algorithm (Xing et al., 2003), one never needs to compute locations but only relative distances.⁵ We thus structure all of our feature functions to take the form of some sort of distance metric and then use all algorithms with the implicit embedding technique. The choice of representation is an important one and a better representation is likely to lead to better performance, especially in the case where the features employed are not amenable to our factorial assumption. Nevertheless, results with this simple model are quite strong, comparative to the other baseline models, and little effort was required to “make the features work” in this task. The only slight complication is that the distances need to be defined so that large distance is correlated with different class, rather than the other way around—this is a problem not faced in conditional or discriminative models such as those of McCallum and Wellner (2004) and Finley and Joachims (2005).

7.3 Appropriateness of DP Prior

Before presenting results on these four data sets, we evaluated whether the assumption that the underlying data distribution comes from a Dirichlet process is reasonable. To do so, we estimated the α parameter for each data set as described in Section 4.3 and computed for each data set size

5. For instance, one of the common calculations is to compute the distances between the means of two subsets of the data, $\{a_i\}_{i=1}^I$ and $\{b_j\}_{j=1}^J$. This can be computed as

$$\left\| \frac{1}{I} \sum_{i=1}^I a_i - \frac{1}{J} \sum_{j=1}^J b_j \right\|^2 = \frac{1}{IJ} \sum_{i=1}^I \sum_{j=1}^J \|a_i - b_j\|^2 - \frac{1}{I^2} \sum_{i=1}^I \sum_{i'=i+1}^I \|a_i - a_{i'}\|^2 - \frac{1}{J^2} \sum_{j=1}^J \sum_{j'=j+1}^J \|b_j - b_{j'}\|^2.$$

The other relevant computations can be done similarly, and the generalization to multidimensional inputs is straightforward.

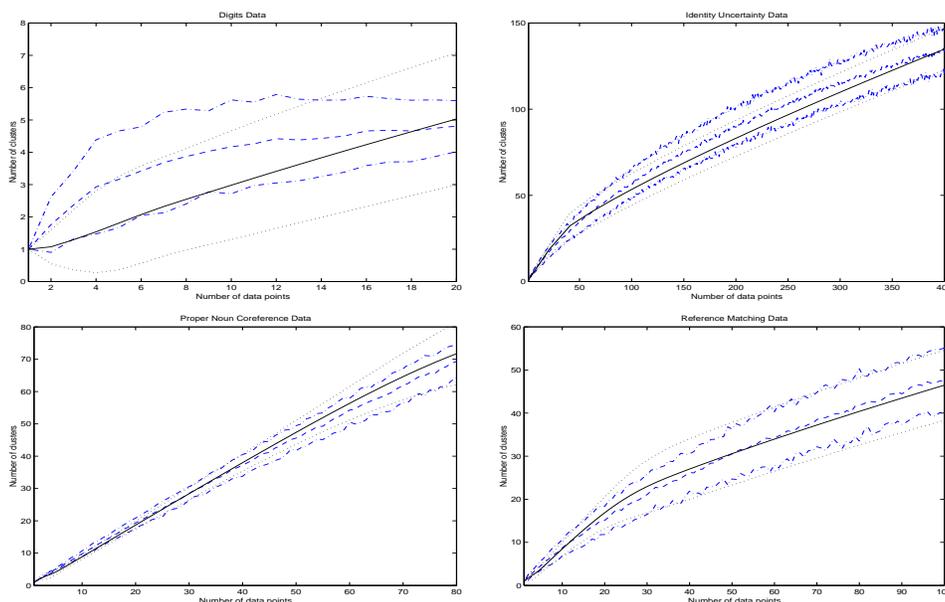


Figure 3: Number of data points by expected number of clusters for the four data sets. The solid black line is the expected number according to the Dirichlet process (dotted black lines are two standard deviations); the dashed blue line is the empirical expected number (dash-dotted blue lines are two standard deviations).

N the expected number of classes K according to the DP (as well as its standard deviation). For each N , we also computed—through resampling—the *empirical* expected value of K according to the data set and its standard deviation. We have plotted these curves for each data set in Figure 3. As we can see from this figure, the DP is an excellent match for most of the data sets, except for the digits data, where the match is rather poor (though the expectations always fall within two standard deviations). Better fits could be obtained using a more complex prior, such as the two-parameter Poisson-Dirichlet process, but we believe that for these tasks, the standard DP is sufficient.

7.3.1 DIGITS DATA

Our first data set is adapted from the USPS digits database (1987), originally a test set of multiclass classification in the vision domain. In order to treat it as a supervised clustering problem, we randomly selected five of the ten digits as the “training data” and use the remaining five as “test data.” The digits used for training are $\{1, 3, 5, 8, 9\}$ and those used for testing are $\{0, 2, 4, 6, 7\}$. The idea is that upon seeing only the digits $\{1, 3, 5, 8, 9\}$, a supervised clustering model should have learned enough about the structure of digits to be able to separate the digits $\{0, 2, 4, 6, 7\}$, even though it has seen none of them (of course, it will not be able to label them).

In order to more closely mimic the fact that in real world data the clusters are rarely equally sized, we artificially “imbalanced” both the training and test data, so that there were between 30 and 300 examples of each of the digits. The digits are 8×8 blocks of pixel intensities, which in all cases are centered and scaled to have unit variance along each dimension. We run ten chains of ten thousand iterations each of the inference algorithm from Figure 2. Each chain of model 1 required

System	RI	P	R	F	CED	NES	VI	NVI
COARSE	.229	.229	1.00	.372	.725	.275	1.525	.765
FINE	.771	1.00	.000	.000	1.00	.008	4.975	.235
K-MEANS	.760	.481	.656	.555	.350	.412	1.446	.778
CDP	.886	.970	.016	.031	1.00	.000	4.237	.241
BINARY	.921	.730	.497	.592	.455	.372	1.193	.805
XING-K	.821	.610	.605	.608	.245	.478	1.165	.821
SCDP-1	.848	.668	.664	.666	.239	.483	1.176	.819
SCDP-2	.854	.692	.659	.675	.227	.538	1.118	.828
SCDP-3	.889	.761	.751	.756	.158	.710	0.791	.878

Table 1: Results on the digits data.

about 40 minutes to complete; model 2’s chains required approximately one hour and the chains from model 3 required 5 hours to complete.

The results of the systems on the digits data are shown in Table 1. There are several things to note in these results. Somewhat surprising is the relatively poor performance of the BINARY model. Indeed, this model barely does better than plain K-means, which completely ignores the training data. Learning a distance metric, in the XING-K system, improves results over standard K-means, and also performs better than the binary classifier. The ordering of performance of our model, compared to the learned distance metric, varies by which metric we believe. According to **F-score** and **NES**, our model is universally better; however, according to **VI** and **NVI**, the distance metric method outperforms our model 1, but not models 2 and 3. Finally, on this data, our untrained model, CDP performs quite poorly, and makes far too many clusters.

7.3.2 IDENTITY UNCERTAINTY DATA

The second data that set we apply our algorithm to is based on the real world problem of *identity uncertainty* or *entity integration*. The data used in the experiment is mined from the dblp bibliography server.⁶ Each “publication” in the data is a computer science researcher and each “reference” is a name occurring in a reference. There are a total of 1382 elements in the data, corresponding to 328 total entities (all labeled). We use 1004 instances (225 entities) as training data and the rest (378 instances and 103 entities that do not occur in training) as testing data.

The (pairwise) features we use for this data set are the following: string edit distance between the two first names; string edit distance between the two last names; string edit distance between the full names; Euclidean distance between the publication years; Euclidean distance between the number of publications (in our data) published in those years; string edit distance between conference names; Euclidean distance between the number of publications published in those conferences; and the number of coauthors with normalized string edit distance less than 0.1. We ran fifty chains of ten thousand iterations each. One chain for Models 1 and 2 required approximately one day to complete, while Model 3 took approximately 3 days per chain.

We introduce an additional baseline for this data set that groups person names with identical last names and identical first initials. This baseline is denoted NAMEMATCH. The results of the systems on the identity uncertainty data are shown in Table 2. The trend of results here largely agrees with

6. Thanks to Anhai Doan, Hui Fang and Rishi R. Sinha for making this available, see <http://anhai.cs.uiuc.edu/archive/domains/researchers.html> for further information.

System	RI	P	R	F	CED	NES	VI	NVI
COARSE	.079	.079	1.00	.147	.749	.251	3.589	.395
FINE	.921	1.00	.000	.000	.000	.273	2.345	.605
NAMEMATCH	.933	.545	1.00	.706	.405	.595	1.252	.789
K-MEANS	.912	.451	.510	.479	.341	.373	1.919	.677
CDP	.913	.480	.452	.466	.355	.360	2.031	.658
BINARY	.855	.753	.801	.776	.389	.553	1.193	.808
XING-K	.916	.467	.423	.444	.378	.304	2.112	.644
SCDP-1	.963	.764	.786	.775	.127	.761	0.806	.864
SCDP-2	.971	.820	.814	.817	.111	.796	0.669	.887
SCDP-3	.982	.875	.913	.894	.066	.876	0.423	.929

Table 2: Results on the identity uncertainty data.

that of the digits data, in which our models 2 and 3 outperform the baseline systems. However, in this case, running the distance-metric learning algorithm actually hurts the results. This is perhaps because our data does not live in Euclidean space, and hence the optimization performed in learning the distance metric is not run under the proper conditions.

In this data, according to the **F-score**, the binary classifier outperforms our model 1 (though our models 2 and 3 outperform the binary classifier). However, according to both the edit distance metrics and the information metrics, our models all outperform the binary classifier. This data also provides a good example of the deficiencies of the rand index: according to the RI, the FINE system outperforms all of: K-MEANS, CDP, BINARY and XING-K. Note also in this data that none of the unsupervised models are able to outperform the NAMEMATCH baseline system (and neither does the XING-K system).

7.3.3 PROPER NOUN COREFERENCE DATA

The third set of data on which we evaluate is a subtask of the coreference task, namely, coreference of proper nouns (e.g., “George Bush” \leftrightarrow “President Bush” \leftrightarrow “Bush” $\not\leftrightarrow$ “President Clinton”). This subtask is significantly simpler than the full task, since one need not identify coreference between pronouns and proper nouns (“he” \leftrightarrow “George Bush”), nor proper nouns and definite descriptions (“George Bush” \leftrightarrow “the President”). This task has previously been used as a benchmark by McCallum and Wellner (2004). We use a partition of the ACE 2004 broadcast news and newswire training corpus as the training and test data. This totals 280 documents for training data and 59 documents for test data. The training data consists of 5613 mentions of entities, corresponding to a total of 3100 different entities; the test data contains 950 mentions corresponding to 523 entities.

As features we use string edit distance, string edit distance on the heads (final word), the length of the longest common substring, the length of the longest common subsequence, and the string edit distance between the abbreviations of both terms. For computing this final term, we first map words sequences like “George W. Bush” to “GWB” and leave sequences that already look like abbreviations (eg., “IBM”) alone; we then compute string edit distance between these pairs. For this data set, we ran fifty chains for ten thousand iterations. Models 1 and 2 completed in about three days and Model 3 completed in one week.

As an additional baseline, we cluster mentions with the same head word (final word); this is denoted SAMEHEAD. The results of the systems on the coreference data are shown in Table 3. As a point of comparison, McCallum and Wellner (2004) report an **F-score** of .931 on this task, using

System	RI	P	R	F	CED	NES	VI	NVI
COARSE	.003	.003	1.00	.006	.978	.021	5.950	.132
FINE	.997	1.00	.000	.000	1.00	.551	0.906	.868
SAMEHEAD	.999	.965	.899	.931	.019	.933	0.123	.982
K-MEANS	.994	.297	.773	.429	.391	.524	1.059	.846
CDP	.995	.273	.384	.319	.352	.418	1.265	.815
BINARY	.999	.900	.893	.896	.040	.936	0.141	.979
XING-K	.998	.489	.802	.608	.308	.599	0.911	.883
SCDP-1	.996	.409	.497	.449	.261	.564	0.938	.863
SCDP-2	.997	.596	.717	.651	.203	.682	0.654	.904
SCDP-3	.999	.804	.882	.841	.083	.861	0.284	.958

Table 3: Results on the proper noun coreference data.

a graph partition strategy, with weights trained using a perceptron-style algorithm. Our binary classification model achieve a slightly lower **F-score** of .896. Neither of the unsupervised algorithms perform very well on this data, but in this data, the trained distance metric performs better than standard K-means.

Overall the binary classifier is the best of the learned systems, achieving an F of .896, a NES of .936 and a normalized variation of information of .979 (compared to our best scores of .841, .861 and .958, respectively). However, even the binary classifier is outperformed along all metrics by the simple baseline that matches on the final word, SAMEHEAD, which achieves scores of .931, .933 and .982 for the three overall metrics. The .931 **F-score** is, incidentally, the same number reported by McCallum and Wellner (2004) (though their choice of training/test division is likely different from ours). Overall, however, based on this data, it seems reasonable to say that one might be better served writing a dozen more rules to capture notions of abbreviation, post-modification, and a few other simple phenomena to handle the proper noun coreference task, rather than try to learn a model from data.⁷

7.3.4 REFERENCE MATCHING DATA

Lastly, we perform evaluation on the Cora reference matching data set McCallum et al. (2000).⁸ This data consists of all references from their collection to publications by Michael Kearns, Robert Schapire and Yoav Freund. There are 1916 references and 121 publications. In the original publication, McCallum et al. treated this as a pure clustering task. In order to view it as a supervised clustering task, we treat the labeled data for two of these authors as training data, using the last author as testing data (performing the segmentation this way is more realistic than random selection, and also serves to strengthen the point that the training and testing data are largely unrelated).

We use the same feature set as in the identity uncertainty evaluation, with the exception that the first two features become the string edit distance between the publication names and the string edit distance between the primary author names, respectively. Note that this data is significantly noisier than the data used in the previous section: there are errors on the labeling of the fields. We again ran fifty chains for ten thousand iterations; the chains for Models 1 and 2 took one day and Model 3 took three days.

7. Of course, the proper-noun coreference task is the easiest subtask of full coreference resolution, where empirical results have shown learned systems are able to outperform rule-based systems.

8. Thanks to Andrew McCallum for making this data available.

System	RI	P	R	F	CED	NES	VI	NVI
COARSE	.118	.118	1.00	.205	.745	.255	2.977	.538
FINE	.882	1.00	.000	.000	.000	.105	3.456	.462
K-MEANS	.862	.407	.461	.433	.392	.240	2.655	.577
CDP	.850	.353	.379	.365	.449	.125	2.948	.531
BINARY	.936	.804	.616	.686	.107	.721	0.762	.881
XING-K	.855	.369	.384	.377	.411	.180	2.807	.552
SCDP-1	.892	.529	.507	.518	.319	.372	2.237	.643
SCDP-2	.934	.696	.741	.718	.184	.641	1.382	.780
SCDP-3	.952	.794	.782	.788	.125	.757	0.957	.847

Table 4: Results on the reference matching data.

	BINARY	XING-K	SCDP-1	SCDP-2	SCDP-3
Digits	.592	.608	.666	.675	.756
Identity Uncertainty	.776	.444	.775	.817	.894
Proper Noun Coreference	.896	.608	.449	.651	.841
Reference Matching	.686	.377	.518	.718	.788

Table 5: Summary of F-scores of the learning systems on all four data sets.

The results of the systems on the reference matching data are shown in Table 4. In this data, the unsupervised algorithms perform quite poorly, in comparison to the systems that make use of the training data. Again, as in the identity uncertainty data, we see that learning a distance metric can hurt performance (at least according to **F-score**; with respect to edit score and normalized **VI**, it seems to help, but only marginally so).

According to **F-score**, the binary classifier on this data outperforms our model 1, though our models 2 and 3 are able to outperform the binary classifier system. In terms of edit score, the binary system outperforms all of our models, except for our model 3, which is able to do slightly better (.757 versus .721). In terms of **NVI**, the binary classifier beats all of our models, even model 3, where it achieves an **NVI** of .881 and we only achieve .847.

7.4 Summary of Results

We have summarized the results of the five learning systems in Table 5 by listing only their final **F-score**. Across all data sets, we consistently see that the supervised approaches outperform the unsupervised approaches, which is not a terribly surprising finding. Additionally, the standard K-means algorithm always outperformed our CDP model (the unsupervised version of our model). In two of the data sets (digits and proper noun coreference), the learned distance metric (XING-K) achieved superior performance to standard K-means, but in the other two data sets, learning the distance metric hurt. In those cases, we attribute this loss in performance to the fact that the algorithm was not operating on truly Euclidean data.

Comparing our models against each other, of our models, model 1 is the poorest performer, followed by model 2, and model 3 is the best. Our models also tend to have higher precision than recall, which suggests that they create too many clusters. One could potentially reduce this by cross-validating on **F-score** to adjust the α^p parameter to attain a balanced precision/recall, but one strong point of Bayesian models is that no cross-validation is necessary.

Our model 3 was able to outperform the binary classification model in most metrics on most data sets, but not always. It tended to consistently outperform the binary classifier in terms of **F-score**, but in terms of **NES** and **NVI**, the binary classifier was better on the reference matching data. On the proper noun coreference data, our model was unable to match the performance of the binary classifier, but both performed more poorly than the simple head-matching baseline system, suggesting that future work on this subtask is perhaps best handled by rules, rather than learning. On the other data sets (digits and identity uncertainty), our models 2 and 3 consistently outperformed the binary classification model.

8. Discussion

In this paper, we have presented a Bayesian model for the supervised clustering problem. We have dealt with the difficulty of defining a prior over a potentially infinite set by appealing to the Dirichlet process prior. We have introduced the concept of a “reference type” as a mechanism for representing the aspects of the data that are general to the entire data set—essentially allowing for the supervision. Like any generative Bayesian classification model, our framework requires the specification of the data generating distribution, which we have denoted F . In general, the F distribution is problem-specific, but we have presented a generic parameterization when F is a Gaussian distribution.

In all but trivial cases, exact evaluation of the posterior distribution of the class variables in our model is intractable. We have presented MCMC-based sampling algorithms that are able to overcome this intractability. Unlike deterministic approximation techniques (such as variational or mean-field inference, or expectation propagation), the MCMC methods are able to perform even when non-conjugate priors are employed. We have presented sampling algorithms for a full Bayesian treatment of the problem.

Experimentally, under the Gaussian assumption our initial model is unable to separate classes well. To fix this problem, we introduced two subsequent models. The first modification we make is to use the references to adjust the parameterization of the prior over the reference types (model 2). This enables the use of a sampling procedure that is essentially as efficient as that used in the original model (model 1). The other modification we employ is to condition the choice of the reference types on the references (model 3). Unfortunately, in this model, the distributions over the reference types and the references are no longer conjugate to the data generating distribution, so a less efficient Gibbs sampler must be employed to perform inference (in general, a single iteration of the non-conjugate model is approximately 10 times slower than one iteration of the conjugate model).

In a systematic comparison on four data sets against both supervised and unsupervised models, we have demonstrated that our model is typically able to attain a higher level of performance than other models (see Section 7.4 for a summary of the experimental results). Full Bayesian inference (similar to transduction) has an advantage over the more standard training/prediction phases: the test data has no influence on the reference types.

The largest weakness of our model is its generative nature and the potential difficulty of specifying a good distribution F that fits the data and results in tractable inference. The Gaussian parameterization seems general, experimentally, but in order to maintain tractability, we had to assume that the covariance matrix was diagonal: this is essentially the same as making a naïve Bayes assumption on the features. For discrete data, using a multinomial/Dirichlet pair or binomial/beta pair instead of the normal/gamma pair might be more natural and would lead to nearly the same inference.

However, like other generative models, it is likely that our model would be struck with the curse of dimensionality for any large number of highly correlated features. The generative story employed by Pasula et al. (2003) is clearly superior to our—largely unmotivated—Gaussian assumption; it would be very interesting to incorporate their generative story into our “ F ” distribution, hopefully to obtain the benefits of both models.

Clearly, scalability is also an issue for our model. The most computationally intensive run of our model with the application of Model 3 to the proper noun coreference data, which required roughly one CPU *year* to perform. This is not to say that, for instance, the binary classification scheme was enormously efficient: training a cross-validated SVM on this data set took approximately one CPU month to perform, though this could be improved by not rerunning the SVM learning for each fold. Nevertheless, our approach is still roughly ten times slower. However, there are several methods that one can employ to improve the speed of the model, especially if we wish to scale the model up to larger data sets. For instance, employing the canopy method described by McCallum et al. (2000) and only considering drawing the c indicator variables from appropriate canopies would drastically improve the efficiency of our model, provided the canopies were sufficiently small. Furthermore, in the cases of Models 1 and 2, since conjugate priors *are* used, one could employ a more efficient sampling scheme, similar to the Metropolis-Hastings algorithm suggested by Xing et al. (2004) or the split-merge proposals suggested by Jain and Neal (2003). Nevertheless, MCMC algorithms are notoriously slow and experiments employing variational or EP methods for the conjugate models might also improve performance (Blei and Jordan, 2005; Minka and Ghahramani, 2004).

Our model is also similar to a distance metric learning algorithm. Under the Gaussian assumption, the reference types become covariance matrices, which—when there is only one reference type—can be interpreted as a transform on the data. However, when there is more than one reference type, or in the case of full Bayesian inference, the sorts of data distributions accounted for by our model are more general than in the standard metric learning scenario.⁹

We believe future research in the context of the framework described in this paper can proceed along several dimensions. The most obvious would be the integration of more domain-specific information in the data generating distribution F . One might be also able to achieve a similar effect by investigating the interaction of our model with various unsupervised embedding techniques (kPCA, LLE, MDS, etc.). We have performed preliminary investigations using kPCA (using the standard string kernel) and LLE combined with K-means as well as K-means and distance-metric learning and have found that performance is substantially worse than the results presented in this paper. A final potential avenue for future work would be to attempt to combine the power of our model with the ability to incorporate arbitrary features found in conditional models, like that of McCallum and Wellner (2004). Such an integration would be technically challenging, but would likely result in a more appropriate, general model.

Finally, to foster further research in the supervised clustering problem, we have contributed our data sets and scoring software to the RIDDLE data repository, <http://www.cs.utexas.edu/users/ml/riddle/>, maintained by Mikhail Bilenko.

9. Consider, for instance, a two dimensional Euclidean space where the clusters are axis-aligned pluses. Our model learns two “reference types” for this data: one aligned with each axis, and, for data that is reasonably separated, is able to correctly classify most test data. On the other hand, a metric learning algorithm cannot perform any linear transformation on the data that will result in “better looking” clusters.

Acknowledgments

The authors would like to thank Aaron D’Souza for several helpful discussions of the Dirichlet process. We would also like to thank the anonymous reviewers of an earlier draft of this paper, whose advice improved this paper dramatically, as well as the three anonymous reviewers of the current version of this paper who contributed greatly to its clarity and content. Some of the computations described in this work were made possible by the High Performance Computing Center at the University of Southern California. This work was partially supported by DARPA-ITO grant N66001-00-1-9814, NSF grant IIS-0097846, and a USC Dean Fellowship to Hal Daumé III.

Appendix A. The Dirichlet Process

The formal definition of the Dirichlet process is as follows. Let (X, Ω) be a measurable space and let μ be a measure (unnormalized density) on this space that is finite, additive, non-negative and non-null. We say that a random probability measure P^μ on (X, Ω) is a *Dirichlet process* with parameter μ under the following condition: whenever $\{B_1, \dots, B_K\}$ is a measurable partition of Ω (i.e., each $\mu(B_k) > 0$ for all k), then the joint distribution of random probabilities $(P^\mu(B_1), \dots, P^\mu(B_K))$ is distributed according to $\mathcal{Dir}(\mu(B_1), \dots, \mu(B_K))$, where \mathcal{Dir} denotes the standard Dirichlet distribution (Ferguson, 1973, 1974). In words: P^μ is a Dirichlet process if it behaves as if it were a Dirichlet distribution on any finite partition of the original space.

It is typically useful to write $\mu = \alpha G_0$, where $\alpha = \int_{\Omega} d\mu$ and $G_0 = \mu/\alpha$, so that G_0 is a density. In this case we refer to G_0 as the *base distribution* or the *mean distribution* of the DP, and α as the *precision*, or *scale parameter*.

Two fundamental results regarding the DP that are important to us are: (1) observations from a DP are discrete (with probability one) and (2) if P^μ is a DP with parameter μ , then the conditional distribution of P^μ given a sample X_1, \dots, X_N is a DP with parameter $P^\mu + \sum_{n=1}^N \delta_{X_n}$, where δ_X is a point mass concentrated at X (Ferguson, 1974). The final useful fact is a correspondence between the DP and Pòlya Urns, described by Blackwell and MacQueen (1973). In the Pòlya Urn construction, we consider the situation of an urn from which we draw balls. Initially the urn contains a single black ball. At any time step, we draw a ball x from the urn. If x is black (as it must be on the first draw), we put x back into the urn and also add a ball of a brand new color. If x was not black, we put x back into the urn and also put in an additional ball of the same color. The pattern of draws from such an urn describes draws from a DP (with $\alpha = 1$). In this scheme, we can see that there is a clustering effect in this model: as more balls of one color (say, blue) are drawn, the number of blue balls in the urn increases, so the probability of drawing a blue ball in the next iteration is higher. However, regardless of how many balls there are in the urn, there is always some probability the black ball (i.e., a ball of a new color) is drawn. This relative probability is controlled by the precision parameter α . For low α , there will be few colors and for high α , there will be many colors. The appropriateness of such a prior depends on one’s prior intuitions about the problem; more flexible similar priors are given in terms of exchangeable probability partition functions, including a simple two-parameter extension of the DP, by Pitman (1996).

As noted by Ferguson (1983), the discreteness of observations from the DP means that observations from the distributions drawn from a DP can be viewed as countably infinite mixtures. This can be seen directly by considering a model that first draws a distribution G from a DP with parameter αG_0 and then draws observations θ_1, \dots from G . In such a model, one can analytically integrate

out G to obtain the following conditional distributions from the observations θ_n (Blackwell and MacQueen, 1973; Ferguson, 1983):

$$\theta_{n+1} \mid \theta_1, \dots, \theta_n \sim \frac{\alpha}{n + \alpha} G_0 + \frac{1}{n + \alpha} \sum_{i=1}^n \delta_{\theta_i}.$$

Thus, the $n + 1$ st data point is drawn with probability proportional to α from the base distribution G_0 , and is exactly equal to a previously drawn θ_i with probability proportional to $\sum_{j=1}^n \delta_{\theta_i=\theta_j}$. This characterization leads to a straightforward implementation of a Gibbs sampler. It also enables one to show that the posterior density of a DP with parameter μ after observing N observations $\theta_1, \dots, \theta_N$ is again a DP with parameter $\mu + \sum_{n=1}^N \delta_{\theta_n}$ (Ferguson, 1973).

Appendix B. Sampling the Precision Parameter

West (1992) describes a method of sampling the precision parameter α for a DP mixture model. Placing a $Gam(a, b)$ prior over α , when n (the number of observations) and k (the number of unique mixture components) are known, one first samples an intermediary value x by a beta distribution $x^\alpha(1-x)^{n-1}$, where α is the previous value for the precision parameter. Given this random variable x , one resamples α according to a mixture of two gamma densities:

$$\pi_x Gam(a + k, b - \log x) + (1 - \pi_x) Gam(a + k - 1, b - \log x),$$

where π_x is the solution to $\pi_x/(1 - \pi_x) = (a + k - 1)/[n(b - \log x)]$. To extend this method to the case with multiple n and k , we first recall the result of Antoniak (1974), which states that the prior distribution of k given α and n is given by

$$p(k \mid \alpha, n) = c_n(k) n! \alpha^k \frac{\Gamma(\alpha)}{\Gamma(\alpha + n)}.$$

Here, $c_n(k) \propto |S_n^{(k)}|$, a Stirling number of the first kind, does not depend on α . Placing a gamma prior on α with shape parameter a and scale parameter b , we obtain the posterior distribution of α given all the n_m, k_m as

$$\begin{aligned} p(\alpha \mid x, k, n) &\propto e^{-b\alpha} \alpha^{a-1} \prod_{m=1}^M \alpha^{k_m-1} (\alpha + n_m) x_m^\alpha (1 - x_m)^{n_m-1} \\ &\propto \alpha^{a-M-1+\sum_{m=1}^M k_m} e^{-\alpha(b-\log \prod_{m=1}^M x_m)} \prod_{m=1}^M (\alpha + n_m). \end{aligned} \quad (4)$$

The product in Equation (4) can be written as the sum over a vector of binary indicator variables i of length M , which gives us

$$\alpha \mid x, k, n \sim \sum_{i \in 2^M} \rho_i Gam \left(a - M + \sum_{m=1}^M k_m + i_m, b - \log \prod_{m=1}^M x_m \right). \quad (5)$$

Where, writing \hat{a} to denote the value $a - M - 1 + \sum_{m=1}^M k_m$ and \hat{b} to denote $b - \log \prod_{m=1}^M x_m$, the mixing weights ρ_i are defined by

$$\rho_i = \frac{1}{Z} \Gamma \left(\hat{a} + \sum_{m=1}^M i_m \right) \prod_{m=1}^M (n_m \hat{b})^{1-i_m}. \tag{6}$$

To see the correctness of this derivation, consider a given i . There are $\sum i_m$ choices of α , corresponding to the $\sum i_m$ in the shape parameter for the posterior gamma distribution in Equation (5). For each of these, the constant from the gamma distribution is decreased by a factor of $\Gamma(\hat{a} + \sum i_m) / \Gamma(\hat{a})$; compensating for this results in the first term above (with the bottom half omitted since it is just a constant). Additionally, each term for which $i_m = 0$ means that n_m was chosen (instead of α), so a factor of $n_m = n_m^{1-i_m}$ needs to be included. Finally, when the shape parameter of the gamma distribution increases by 1 for each $i_m = 1$, the constant of proportionality for the gamma distribution increases by a factor of $b - \log \prod x_m$, which is compensated for by the last term above.

Similarly, we can obtain a marginal distribution for each x_m conditional on α and k as:

$$x_m \mid \alpha, n_m, k_m \propto x_m^\alpha (1 - x_m)^{n_m - 1} \sim \text{Bet}(\alpha + 1, n_m) \tag{7}$$

In order to sample α , we first sample x by a sequence of m beta distributions according to Equation (7), conditioned on the current value of α and n . Then, given these values of x , we sample a new value of α from a mixture of gammas defined in Equation (5), conditional on the newly sampled x , with weights defined in Equation (6). In the latter step, we simply select an $i \in 2^M$ according to the probability density ρ_i and then sample a value from the corresponding gamma distribution.

Unfortunately, in all but trivial cases, M is large and so computing ρ_i directly for all such i requires an exponential amount of time (in M). Thus, instead of computing the ρ s directly, we sample for them, effectively computing the constant Z through standard MCMC techniques. To perform the actual sampling from 2^M , we employ a Gibbs sampler. Each iteration of the Gibbs sampler cycles through each of the M values of i and replaces i_m with a new value, sampled according to its posterior, conditional on $i_{-m} = \langle i_l \mid 1 \leq l \leq M, l \neq m \rangle$. The derivation of this posterior is straightforward:

$$i_m = 1 \mid i_{-m} = \frac{\hat{a} + \sum_{m' \neq m} i_{m'}}{\hat{a} + \sum_{m' \neq m} i_{m'} + n_m \hat{b}}. \tag{8}$$

Putting it all together, we sample a new value of α by first sampling a vector x , where each x_m is sampled according to Equation (7). Then, we sample R -many $i^{(r)}$ s using the Gibbs sampler with update given by Equation (8); finally selecting one of the $i^{(r)}$ according to its empirical density. Finally, given this i and the x_m s, we sample a new value for α by a gamma distribution according to Equation (5). We have found that for modest $M < 100$, $n_m < 1000$ and $k_m < 500$, such a chain converges in roughly 50 iterations. In practice, we run it for 200 iterations to be safe.

References

Charles E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, November 1974.

- Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning distance functions using equivalence relations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.
- Aharon Bar-Hillel and Daphna Weinshall. Learning with equivalence constraints and the relation to multiclass learning. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2003.
- Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 42–49, 2003.
- Matt Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.
- David Blackwell and James B. MacQueen. Ferguson distributions via Pòlya urn schemes. *The Annals of Statistics*, 1(2):353–355, March 1973.
- David Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144, August 2005.
- Peter Carbonetto, Jacek Kisiński, Nando de Freitas, and David Poole. Nonparametric bayesian logic. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, July 2005.
- William Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD*, 2002.
- Anhai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontology matching: A machine learning approach. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Information Systems*, pages 397–416. Springer-Verlag, 2004.
- Michael D. Escobar. Estimating normal means with a Dirichlet process prior. *Journal of the American Statistical Association (JASA)*, 89(425):268–277, March 1994.
- Thomas S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, March 1973.
- Thomas S. Ferguson. Prior distributions on spaces of probability measures. *The Annals of Statistics*, 2(4):615–629, July 1974.
- Thomas S. Ferguson. Bayesian density estimation by mixtures of normal distribution. In H. Rizvi and J. Rustagi, editors, *Recent Advances in Statistics*, pages 287–303. Academic Press, 1983.
- Thomas Finley and Thorsten Joachims. Supervised clustering with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2005.
- Hermant Ishwaran and Lancelot F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association (JASA)*, 96(453):161–173, March 2001.

- Sonia Jain and Radford M. Neal. A split-merge markov chain Monte Carlo procedure for the Dirichlet process mixture model. Technical Report 2003, University of Toronto, Department of Statistics, 2003.
- Toshihiro Kamishima and Fumio Motoyoshi. Learning from cluster examples. *Machine Learning (ML)*, pages 199–233, 2003.
- Steven N. MacEachern and Peter Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphical Statistics (JCGS)*, 7:223–238, 1998.
- Andrew McCallum, Kamal Nigam, and Lyle Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.
- Andrew McCallum and Ben Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.
- Marina Meila. Comparing clusterings. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2003.
- Thomas Minka and Zoubin Ghahramani. Expectation propagation for infinite mixtures. In *NIPS Workshop on Nonparametric Bayesian Methods and Infinite Models*, 2004.
- Alvaro E. Monge and Charles Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *KDD*, 1997.
- Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Technical Report 9815, University of Toronto, September 1998.
- Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, 2002.
- Patrick Pantel. *Clustering by Committee*. PhD thesis, University of Alberta, 2003.
- Hanna Pasula, Bhaskara Marthi, Brian Milch, Stuart Russell, and Ilya Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.
- Jim Pitman. Some developments of the Blackwell-MacQueen urn scheme. In *Statistics, Probability and Game Theory; Papers in honor of David Blackwell Lecture Notes*, Monograph Series 30: 245–267, 1996.
- W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association (JASA)*, 66:846–850, 1971.
- Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521 – 544, 2001.
- Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasmine Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

USPS digits database. United states postal service handwritten zip code database. Made available by the USPS Office of Advanced Technology, 1987.

Mike West. Hyperparameter estimation in Dirichlet process mixture models. *ISDS Discussion Paper #92-A03*, 1992. Duke University.

Eric P. Xing, Andrew Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

Eric P. Xing, Roded Sharan, and Michael I. Jordan. Bayesian haplotype inference via the Dirichlet process. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

Fast Kernel Classifiers with Online and Active Learning

Antoine Bordes

*NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA, and
Ecole Supérieure de Physique et Chimie Industrielles
10 rue Vauquelin
75231 Paris CEDEX 05, France*

ANTOINE.BORDES@BDE.ESPCI.FR

Seyda Ertekin

*The Pennsylvania State University
University Park, PA 16802, USA*

SEYDA@PSU.EDU

Jason Weston

*NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA*

JASONW@NEC-LABS.COM

Léon Bottou

*NEC Laboratories America
4 Independence Way
Princeton, NJ 08540, USA*

LEON@BOTTOU.ORG

Editor: Nello Cristianini

Abstract

Very high dimensional learning systems become theoretically possible when training examples are abundant. The computing cost then becomes the limiting factor. Any efficient learning algorithm should at least take a brief look at each example. But should all examples be given equal attention?

This contribution proposes an empirical answer. We first present an online SVM algorithm based on this premise. LASVM yields competitive misclassification rates after a single pass over the training examples, outspeeding state-of-the-art SVM solvers. Then we show how active example selection can yield faster training, higher accuracies, and simpler models, using only a fraction of the training example labels.

1. Introduction

Electronic computers have vastly enhanced our ability to compute complicated statistical models. Both theory and practice have adapted to take into account the essential compromise between the number of examples and the model capacity (Vapnik, 1998). Cheap, pervasive and networked computers are now enhancing our ability to collect observations to an even greater extent. Data sizes outgrow computer speed. During the last decade, processors became 100 times faster, hard disks became 1000 times bigger.

Very high dimensional learning systems become theoretically possible when training examples are abundant. The computing cost then becomes the limiting factor. Any efficient learning algorithm should at least pay a brief look at each example. But should all training examples be given equal attention?

This contribution proposes an empirical answer:

- Section 2 presents kernel classifiers such as Support Vector Machines (SVM). Kernel classifiers are convenient for our purposes because they clearly express their internal states in terms of subsets of the training examples.
- Section 3 proposes a novel online algorithm, LASVM, which converges to the SVM solution. Experimental evidence on diverse data sets indicates that it reliably reaches competitive accuracies after performing a single pass over the training set. It uses less memory and trains significantly faster than state-of-the-art SVM solvers.
- Section 4 investigates two criteria to select informative training examples at each iteration instead of sequentially processing all examples. Empirical evidence shows that selecting informative examples without making use of the class labels can drastically reduce the training time and produce much more compact classifiers with equivalent or superior accuracy.
- Section 5 discusses the above results and formulates theoretical questions. The simplest question involves the convergence of these algorithms and is addressed by the appendix. Other questions of greater importance remain open.

2. Kernel Classifiers

Early linear classifiers associate classes $y = \pm 1$ to patterns x by first transforming the patterns into feature vectors $\Phi(x)$ and taking the sign of a linear discriminant function:

$$\hat{y}(x) = w' \Phi(x) + b. \quad (1)$$

The parameters w and b are determined by running some learning algorithm on a set of training examples $(x_1, y_1) \cdots (x_n, y_n)$. The feature function Φ is usually hand chosen for each particular problem (Nilsson, 1965).

Aizerman et al. (1964) transform such linear classifiers by leveraging two theorems of the *Reproducing Kernel* theory (Aronszajn, 1950).

The *Representation Theorem* states that many Φ -machine learning algorithms produce parameter vectors w that can be expressed as a linear combinations of the training patterns:

$$w = \sum_{i=1}^n \alpha_i \Phi(x_i).$$

The linear discriminant function (1) can then be written as a *kernel expansion*

$$\hat{y}(x) = \sum_{i=1}^n \alpha_i K(x, x_i) + b, \quad (2)$$

where the *kernel* function $K(x, y)$ represents the dot products $\Phi(x)' \Phi(y)$ in feature space. This expression is most useful when a large fraction of the coefficients α_i are zero. Examples such that $\alpha_i \neq 0$ are then called *Support Vectors*.

Mercer's Theorem precisely states which kernel functions correspond to a dot product for some feature space. Kernel classifiers deal with the kernel function $K(x, y)$ without explicitly using the

corresponding feature function $\Phi(x)$. For instance, the well known *RBF* kernel $K(x, y) = e^{-\gamma\|x-y\|^2}$ defines an implicit feature space of infinite dimension.

Kernel classifiers handle such large feature spaces with the comparatively modest computational costs of the kernel function. On the other hand, kernel classifiers must control the decision function complexity in order to avoid overfitting the training data in such large feature spaces. This can be achieved by keeping the number of support vectors as low as possible (Littlestone and Warmuth, 1986) or by searching decision boundaries that separate the examples with the largest margin (Vapnik and Lerner, 1963; Vapnik, 1998).

2.1 Support Vector Machines

Support Vector Machines were defined by three incremental steps. First, Vapnik and Lerner (1963) propose to construct the *Optimal Hyperplane*, that is, the linear classifier that separates the training examples with the widest margin. Then, Guyon, Boser, and Vapnik (1993) propose to construct the Optimal Hyperplane in the feature space induced by a kernel function. Finally, Cortes and Vapnik (1995) show that noisy problems are best addressed by allowing some examples to violate the margin condition.

Support Vector Machines minimize the following objective function in feature space:

$$\min_{w,b} \|w\|^2 + C \sum_{i=1}^n \xi_i \quad \text{with} \quad \begin{cases} \forall i & y_i \hat{y}(x_i) \geq 1 - \xi_i \\ \forall i & \xi_i \geq 0. \end{cases} \quad (3)$$

For very large values of the hyper-parameter C , this expression minimizes $\|w\|^2$ under the constraint that all training examples are correctly classified with a margin $y_i \hat{y}(x_i)$ greater than 1. Smaller values of C relax this constraint and produce markedly better results on noisy problems (Cortes and Vapnik, 1995).

In practice this is achieved by solving the dual of this convex optimization problem. The coefficients α_i of the SVM kernel expansion (2) are found by defining the dual objective function

$$W(\alpha) = \sum_i \alpha_i y_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \quad (4)$$

and solving the SVM *Quadratic Programming* (QP) problem:

$$\max_{\alpha} W(\alpha) \quad \text{with} \quad \begin{cases} \sum_i \alpha_i = 0 \\ A_i \leq \alpha_i \leq B_i \\ A_i = \min(0, Cy_i) \\ B_i = \max(0, Cy_i). \end{cases} \quad (5)$$

The above formulation slightly deviates from the standard formulation (Cortes and Vapnik, 1995) because it makes the α_i coefficients positive when $y_i = +1$ and negative when $y_i = -1$.

SVMs have been very successful and are very widely used because they reliably deliver state-of-the-art classifiers with minimal tweaking.

Computational Cost of SVMs There are two intuitive lower bounds on the computational cost of any algorithm able to solve the SVM QP problem for arbitrary matrices $K_{ij} = K(x_i, x_j)$.

1. Suppose that an oracle reveals whether $\alpha_i = 0$ or $\alpha_i = \pm C$ for all $i = 1 \dots n$. Computing the remaining $0 < |\alpha_i| < C$ amounts to inverting a matrix of size $R \times R$ where R is the number of support vectors such that $0 < |\alpha_i| < C$. This typically requires a number of operations proportional to R^3 .
2. Simply verifying that a vector α is a solution of the SVM QP problem involves computing the gradients of $W(\alpha)$ and checking the Karush-Kuhn-Tucker optimality conditions (Vapnik, 1998). With n examples and S support vectors, this requires a number of operations proportional to nS .

Few support vectors reach the upper bound C when it gets large. The cost is then dominated by the $R^3 \approx S^3$. Otherwise the term nS is usually larger. The final number of support vectors therefore is the critical component of the computational cost of the SVM QP problem.

Assume that increasingly large sets of training examples are drawn from an unknown distribution $P(x, y)$. Let \mathcal{B} be the error rate achieved by the best decision function (1) for that distribution. When $\mathcal{B} > 0$, Steinwart (2004) shows that the number of support vectors is asymptotically equivalent to $2n\mathcal{B}$. Therefore, regardless of the exact algorithm used, the asymptotical computational cost of solving the SVM QP problem grows at least like n^2 when C is small and n^3 when C gets large. Empirical evidence shows that modern SVM solvers (Chang and Lin, 2001-2004; Collobert and Bengio, 2001) come close to these scaling laws.

Practice however is dominated by the constant factors. When the number of examples grows, the kernel matrix $K_{ij} = K(x_i, x_j)$ becomes very large and cannot be stored in memory. Kernel values must be computed on the fly or retrieved from a cache of often accessed values. When the cost of computing each kernel value is relatively high, the kernel cache hit rate becomes a major component of the cost of solving the SVM QP problem (Joachims, 1999). Larger problems must be addressed by using algorithms that access kernel values with very consistent patterns.

Section 3 proposes an Online SVM algorithm that accesses kernel values very consistently. Because it computes the SVM optimum, this algorithm cannot improve on the n^2 lower bound. Because it is an online algorithm, early stopping strategies might give approximate solutions in much shorter times. Section 4 suggests that this can be achieved by carefully choosing which examples are processed at each iteration.

Before introducing the new Online SVM, let us briefly describe other existing online kernel methods, beginning with the kernel Perceptron.

2.2 Kernel Perceptrons

The earliest kernel classifiers (Aizerman et al., 1964) were derived from the Perceptron algorithm (Rosenblatt, 1958). The decision function (2) is represented by maintaining the set S of the indices i of the support vectors. The bias parameter b remains zero.

Kernel Perceptron

- 1) $S \leftarrow \emptyset$, $b \leftarrow 0$.
- 2) Pick a random example (x_t, y_t)
- 3) Compute $\hat{y}(x_t) = \sum_{i \in S} \alpha_i K(x_t, x_i) + b$
- 4) If $y_t \hat{y}(x_t) \leq 0$ then $S \leftarrow S \cup \{t\}$, $\alpha_t \leftarrow y_t$
- 5) Return to step 2.

Such *Online Learning Algorithms* require very little memory because the examples are processed one by one and can be discarded after being examined.

Iterations such that $y_t \hat{y}(x_t) < 0$ are called *mistakes* because they correspond to patterns misclassified by the perceptron decision boundary. The algorithm then modifies the decision boundary by inserting the misclassified pattern into the kernel expansion. When a solution exists, Novikoff's Theorem (Novikoff, 1962) states that the algorithm converges after a finite number of mistakes, or equivalently after inserting a finite number of support vectors. Noisy data sets are more problematic.

Large Margin Kernel Perceptrons The success of Support Vector Machines has shown that large classification margins were desirable. On the other hand, the Kernel Perceptron (Section 2.2) makes no attempt to achieve large margins because it happily ignores training examples that are very close to being misclassified.

Many authors have proposed to close the gap with online kernel classifiers by providing larger margins. The Averaged Perceptron (Freund and Schapire, 1998) decision rule is the majority vote of all the decision rules obtained after each iteration of the Kernel Perceptron algorithm. This choice provides a bound comparable to those offered in support of SVMs. Other algorithms (Frieß et al., 1998; Gentile, 2001; Li and Long, 2002; Crammer and Singer, 2003) explicitly construct larger margins. These algorithms modify the decision boundary whenever a training example is either misclassified or classified with an insufficient margin. Such examples are then inserted into the kernel expansion with a suitable coefficient. Unfortunately, this change significantly increases the number of mistakes and therefore the number of support vectors. The increased computational cost and the potential overfitting undermines the positive effects of the increased margin.

Kernel Perceptrons with Removal Step This is why Crammer et al. (2004) suggest an additional step for *removing* support vectors from the kernel expansion (2). The Budget Perceptron performs very nicely on relatively clean data sets.

Budget Kernel Perceptron (β, N)

- 1) $\mathcal{S} \leftarrow \emptyset, \quad b \leftarrow 0.$
- 2) Pick a random example (x_t, y_t)
- 3) Compute $\hat{y}(x_t) = \sum_{i \in \mathcal{S}} \alpha_i K(x_t, x_i) + b$
- 4) If $y_t \hat{y}(x_t) \leq \beta$ then,
 - 4a) $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}, \quad \alpha_t \leftarrow y_t$
 - 4b) If $|\mathcal{S}| > N$ then $\mathcal{S} \leftarrow \mathcal{S} - \{\arg \max_{i \in \mathcal{S}} y_i (\hat{y}(x_i) - \alpha_i K(x_i, x_i))\}$
- 5) Return to step 2.

Online kernel classifiers usually experience considerable problems with noisy data sets. Each iteration is likely to cause a mistake because the best achievable misclassification rate for such problems is high. The number of support vectors increases very rapidly and potentially causes overfitting and poor convergence. More sophisticated support vector removal criteria avoid this drawback (Weston et al., 2005). This modified algorithm outperforms all other *online* kernel classifiers on noisy data sets and matches the performance of Support Vector Machines with less support vectors.

3. Online Support Vector Machines

This section proposes a novel online algorithm named LASVM that converges to the SVM solution. This algorithm furthers ideas first presented by Bordes and Bottou (2005). Unlike this previous

work, LASVM relies on the traditional “soft margin” SVM formulation, handles noisy data sets, and is nicely related to the SMO algorithm. Experimental evidence on multiple data sets indicates that it reliably reaches competitive test error rates after performing a single pass over the training set. It uses less memory and trains significantly faster than state-of-the-art SVM solvers.

3.1 Quadratic Programming Solvers for SVMs

Sequential Direction Search Efficient numerical algorithms have been developed to solve the SVM QP problem (5). The best known methods are the Conjugate Gradient method (Vapnik, 1982, pages 359–362) and the Sequential Minimal Optimization (Platt, 1999). Both methods work by making successive searches along well chosen directions.

Each direction search solves the restriction of the SVM problem to the half-line starting from the current vector α and extending along the specified direction u . Such a search yields a new feasible vector $\alpha + \lambda^*u$, where

$$\lambda^* = \arg \max W(\alpha + \lambda u) \quad \text{with} \quad 0 \leq \lambda \leq \phi(\alpha, u). \quad (6)$$

The upper bound $\phi(\alpha, u)$ ensures that $\alpha + \lambda u$ is feasible as well:

$$\phi(\alpha, u) = \min \left\{ \begin{array}{ll} 0 & \text{if } \sum_k u_k \neq 0 \\ (B_i - \alpha_i)/u_i & \text{for all } i \text{ such that } u_i > 0 \\ (A_j - \alpha_j)/u_j & \text{for all } j \text{ such that } u_j < 0. \end{array} \right\} \quad (7)$$

Calculus shows that the optimal value is achieved for

$$\lambda^* = \min \left\{ \phi(\alpha, u), \frac{\sum_i g_i u_i}{\sum_{i,j} u_i u_j K_{ij}} \right\} \quad (8)$$

where $K_{ij} = K(x_i, x_j)$ and $g = (g_1 \dots g_n)$ is the gradient of $W(\alpha)$, and

$$g_k = \frac{\partial W(\alpha)}{\partial \alpha_k} = y_k - \sum_i \alpha_i K(x_i, x_k) = y_k - \hat{y}(x_k) + b. \quad (9)$$

Sequential Minimal Optimization Platt (1999) observes that direction search computations are much faster when the search direction u mostly contains zero coefficients. At least two coefficients are needed to ensure that $\sum_k u_k = 0$. The *Sequential Minimal Optimization* (SMO) algorithm uses search directions whose coefficients are all zero except for a single +1 and a single −1.

Practical implementations of the SMO algorithm (Chang and Lin, 2001-2004; Collobert and Bengio, 2001) usually rely on a small positive tolerance $\tau > 0$. They only select directions u such that $\phi(\alpha, u) > 0$ and $u'g > \tau$. This means that we can move along direction u without immediately reaching a constraint and increase the value of $W(\alpha)$. Such directions are defined by the so-called τ -violating pair (i, j) :

$$(i, j) \text{ is a } \tau\text{-violating pair} \iff \left\{ \begin{array}{l} \alpha_i < B_i \\ \alpha_j > A_j \\ g_i - g_j > \tau. \end{array} \right.$$

SMO Algorithm

- 1) Set $\alpha \leftarrow 0$ and compute the initial gradient g (equation 9)
- 2) Choose a τ -violating pair (i, j) . Stop if no such pair exists.
- 3) $\lambda \leftarrow \min \left\{ \frac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}}, B_i - \alpha_i, \alpha_j - A_j \right\}$
 $\alpha_i \leftarrow \alpha_i + \lambda, \quad \alpha_j \leftarrow \alpha_j - \lambda$
 $g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall s \in \{1 \dots n\}$
- 4) Return to step (2)

The above algorithm does not specify how exactly the τ -violating pairs are chosen. Modern implementations of SMO select the τ -violating pair (i, j) that maximizes the directional gradient $u'g$. This choice was described in the context of Optimal Hyperplanes in both (Vapnik, 1982, pages 362–364) and (Vapnik et al., 1984).

Regardless of how exactly the τ -violating pairs are chosen, Keerthi and Gilbert (2002) assert that the SMO algorithm stops after a finite number of steps. This assertion is correct despite a slight flaw in their final argument (Takahashi and Nishi, 2003).

When SMO stops, no τ -violating pair remain. The corresponding α is called a τ -approximate solution. Proposition 13 in appendix A establishes that such approximate solutions indicate the location of the solution(s) of the SVM QP problem when the tolerance τ become close to zero.

3.2 Online LASVM

This section presents a novel online SVM algorithm named LASVM. There are two ways to view this algorithm. LASVM is an online kernel classifier sporting a support vector removal step: vectors collected in the current kernel expansion can be removed during the online process. LASVM also is a reorganization of the SMO sequential direction searches and, as such, converges to the solution of the SVM QP problem.

Compared to basic kernel perceptrons (Aizerman et al., 1964; Freund and Schapire, 1998), the LASVM algorithm features a removal step and gracefully handles noisy data. Compared to kernel perceptrons with removal steps (Crammer et al., 2004; Weston et al., 2005), LASVM converges to the known SVM solution. Compared to a traditional SVM solver (Platt, 1999; Chang and Lin, 2001–2004; Collobert and Bengio, 2001), LASVM brings the computational benefits and the flexibility of online learning algorithms. Experimental evidence indicates that LASVM matches the SVM accuracy after a single sequential pass over the training examples.

This is achieved by alternating two kinds of direction searches named PROCESS and REPROCESS. Each direction search involves a pair of examples. Direction searches of the PROCESS kind involve at least one example that is not a support vector of the current kernel expansion. They potentially can change the coefficient of this example and make it a support vector. Direction searches of the REPROCESS kind involve two examples that already are support vectors in the current kernel expansion. They potentially can zero the coefficient of one or both support vectors and thus remove them from the kernel expansion.

Building Blocks The LASVM algorithm maintains three essential pieces of information: the set S of potential support vector indices, the coefficients α_i of the current kernel expansion, and the partial derivatives g_i defined in (9). Variables α_i and g_i contain meaningful values when $i \in S$ only.

The coefficient α_i are assumed to be null if $i \notin \mathcal{S}$. On the other hand, set \mathcal{S} might contain a few indices i such that $\alpha_i = 0$.

The two basic operations of the Online LASVM algorithm correspond to steps 2 and 3 of the SMO algorithm. These two operations differ from each other because they have different ways to select τ -violating pairs.

The first operation, PROCESS, attempts to insert example $k \notin \mathcal{S}$ into the set of current support vectors. In the online setting this can be used to process a new example at time t . It first adds example $k \notin \mathcal{S}$ into \mathcal{S} (step 1-2). Then it searches a second example in \mathcal{S} to find the τ -violating pair with maximal gradient (steps 3-4) and performs a direction search (step 5).

LASVM PROCESS(k)

- 1) Bail out if $k \in \mathcal{S}$.
- 2) $\alpha_k \leftarrow 0$, $g_k \leftarrow y_k - \sum_{s \in \mathcal{S}} \alpha_s K_{ks}$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$
- 3) If $y_k = +1$ then
 $i \leftarrow k$, $j \leftarrow \arg \min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$
 else
 $j \leftarrow k$, $i \leftarrow \arg \max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$
- 4) Bail out if (i, j) is not a τ -violating pair.
- 5) $\lambda \leftarrow \min \left\{ \frac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}}, B_i - \alpha_i, \alpha_j - A_j \right\}$
 $\alpha_i \leftarrow \alpha_i + \lambda$, $\alpha_j \leftarrow \alpha_j - \lambda$
 $g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall s \in \mathcal{S}$

The second operation, REPROCESS, removes some elements from \mathcal{S} . It first searches the τ -violating pair of elements of \mathcal{S} with maximal gradient (steps 1-2), and performs a direction search (step 3). Then it removes blatant non support vectors (step 4). Finally it computes two useful quantities: the bias term b of the decision function (2) and the gradient δ of the most τ -violating pair in \mathcal{S} .

LASVM REPROCESS

- 1) $i \leftarrow \arg \max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$
 $j \leftarrow \arg \min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$
- 2) Bail out if (i, j) is not a τ -violating pair.
- 3) $\lambda \leftarrow \min \left\{ \frac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}}, B_i - \alpha_i, \alpha_j - A_j \right\}$
 $\alpha_i \leftarrow \alpha_i + \lambda$, $\alpha_j \leftarrow \alpha_j - \lambda$
 $g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall s \in \mathcal{S}$
- 4) $i \leftarrow \arg \max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$
 $j \leftarrow \arg \min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$
 For all $s \in \mathcal{S}$ such that $\alpha_s = 0$
 If $y_s = -1$ and $g_s \geq g_i$ then $\mathcal{S} = \mathcal{S} - \{s\}$
 If $y_s = +1$ and $g_s \leq g_j$ then $\mathcal{S} = \mathcal{S} - \{s\}$
- 5) $b \leftarrow (g_i + g_j)/2$, $\delta \leftarrow g_i - g_j$

Online LASVM After initializing the state variables (step 1), the Online LASVM algorithm alternates PROCESS and REPROCESS a predefined number of times (step 2). Then it simplifies the kernel expansion by running REPROCESS to remove all τ -violating pairs from the kernel expansion (step 3).

LASVM

1) **Initialization:**

Seed \mathcal{S} with a few examples of each class.
Set $\alpha \leftarrow 0$ and compute the initial gradient g (equation 9)

2) **Online Iterations:**

Repeat a predefined number of times:

- Pick an example k_t
- Run PROCESS(k_t).
- Run REPROCESS once.

3) **Finishing:**

Repeat REPROCESS until $\delta \leq \tau$.

LASVM can be used in the online setup where one is given a continuous stream of fresh random examples. The online iterations process fresh training examples as they come. LASVM can also be used as a stochastic optimization algorithm in the offline setup where the complete training set is available before hand. Each iteration randomly picks an example from the training set.

In practice we run the LASVM online iterations in epochs. Each epoch sequentially visits all the randomly shuffled training examples. After a predefined number P of epochs, we perform the finishing step. A single epoch is consistent with the use of LASVM in the online setup. Multiple epochs are consistent with the use of LASVM as a stochastic optimization algorithm in the offline setup.

Convergence of the Online Iterations Let us first ignore the finishing step (step 3) and assume that online iterations (step 2) are repeated indefinitely. Suppose that there are remaining τ -violating pairs at iteration T .

- a.) If there are τ -violating pairs (i, j) such that $i \in \mathcal{S}$ and $j \in \mathcal{S}$, one of them will be exploited by the next REPROCESS.
- b.) Otherwise, if there are τ -violating pairs (i, j) such that $i \in \mathcal{S}$ or $j \in \mathcal{S}$, each subsequent PROCESS has a chance to exploit one of them. The intervening REPROCESS do nothing because they bail out at step 2.
- c.) Otherwise, all τ -violating pairs involve indices outside \mathcal{S} . Subsequent calls to PROCESS and REPROCESS bail out until we reach a time $t > T$ such that $k_t = i$ and $k_{t+1} = j$ for some τ -violating pair (i, j) . The first PROCESS then inserts i into \mathcal{S} and bails out. The following REPROCESS bails out immediately. Finally the second PROCESS locates pair (i, j) .

This case is not important in practice. There usually is a support vector $s \in \mathcal{S}$ such that $A_s < \alpha_s < B_s$. We can then write $g_i - g_j = (g_i - g_s) + (g_s - g_j) \leq 2\tau$ and conclude that we already have reached a 2τ -approximate solution.

The LASVM online iterations therefore work like the SMO algorithm. Remaining τ -violating pairs is sooner or later exploited by either PROCESS or REPROCESS. As soon as a τ -approximate solution is reached, the algorithm stops updating the coefficients α . Theorem 18 in the appendix gives more precise convergence results for this stochastic algorithm.

The finishing step (step 3) is only useful when one limits the number of online iterations. Running LASVM usually consists in performing a predefined number P of epochs and running the finishing step. Each epoch performs n online iterations by sequentially visiting the randomly shuffled training examples. Empirical evidence suggests indeed that a *single epoch* yields a classifier almost as good as the SVM solution.

Computational Cost of LASVM Both PROCESS and REPROCESS require a number of operations proportional to the number S of support vectors in set \mathcal{S} . Performing P epochs of online iterations requires a number of operations proportional to $nP\bar{S}$. The average number \bar{S} of support vectors scales no more than linearly with n because each online iteration brings at most one new support vector. The asymptotic cost therefore grows like n^2 at most. The finishing step is similar to running a SMO solver on a SVM problem with only S training examples. We recover here the n^2 to n^3 behavior of standard SVM solvers.

Online algorithms access kernel values with a very specific pattern. Most of the kernel values accessed by PROCESS and REPROCESS involve only support vectors from set \mathcal{S} . Only PROCESS on a new example x_{k_t} accesses S fresh kernel values $K(x_{k_t}, x_i)$ for $i \in \mathcal{S}$.

Implementation Details Our LASVM implementation reorders the examples after every PROCESS or REPROCESS to ensure that the current support vectors come first in the reordered list of indices. The kernel cache records truncated rows of the reordered kernel matrix. SVMLight (Joachims, 1999) and LIBSVM (Chang and Lin, 2001-2004) also perform such reorderings, but do so rather infrequently (Joachims, 1999). The reordering overhead is acceptable during the online iterations because the computation of fresh kernel values takes much more time.

Reordering examples during the finishing step was more problematic. We eventually deployed an adaptation of the *shrinking* heuristic (Joachims, 1999) for the finishing step only. The set \mathcal{S} of support vectors is split into an active set \mathcal{S}_a and an inactive set \mathcal{S}_i . All support vectors are initially active. The REPROCESS iterations are restricted to the active set \mathcal{S}_a and do not perform any reordering. About every 1000 iterations, support vectors that hit the boundaries of the box constraints are either removed from the set \mathcal{S} of support vectors or moved from the active set \mathcal{S}_a to the inactive set \mathcal{S}_i . When all τ -violating pairs of the active set are exhausted, the inactive set examples are transferred back into the active set. The process continues as long as the merged set contains τ -violating pairs.

3.3 MNIST Experiments

The Online LASVM was first evaluated on the MNIST¹ handwritten digit data set (Bottou et al., 1994). Computing kernel values for this data set is relatively expensive because it involves dot products of 784 gray level pixel values. In the experiments reported below, all algorithms use the same code for computing kernel values. The ten binary classification tasks consist of separating each digit class from the nine remaining classes. All experiments use RBF kernels with $\gamma = 0.005$

1. This data set is available at <http://yann.lecun.com/exdb/mnist>.

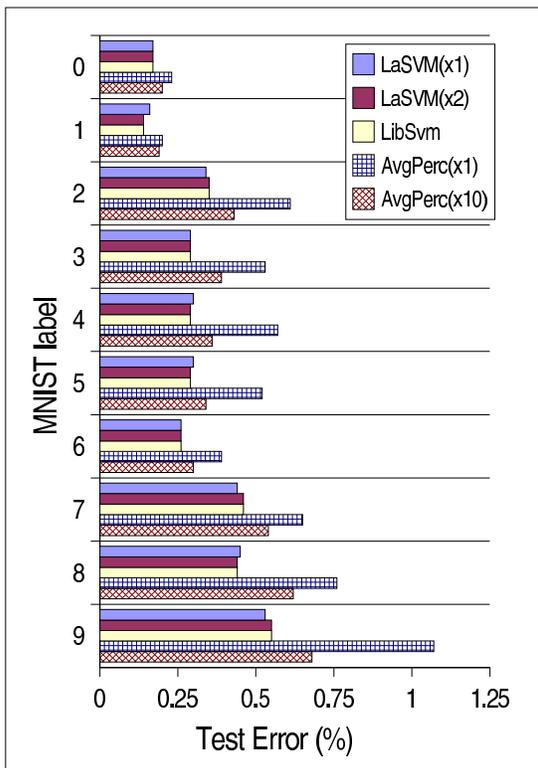


Figure 1: Compared test error rates for the ten MNIST binary classifiers.

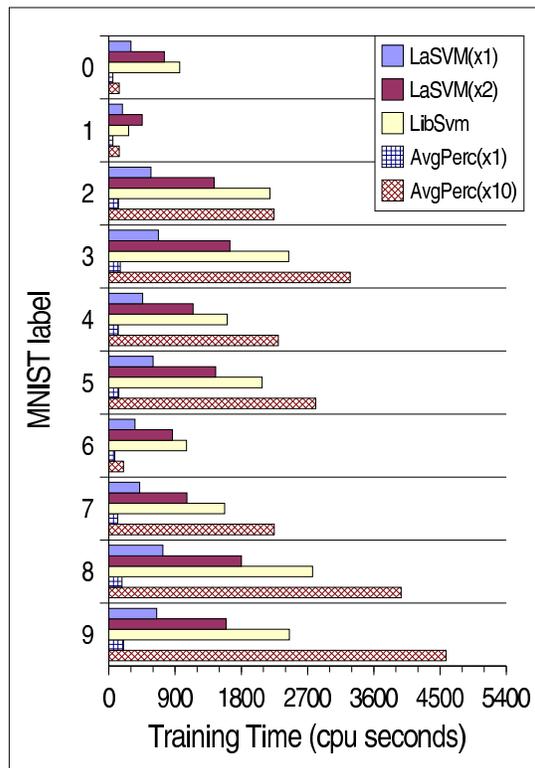


Figure 2: Compared training times for the ten MNIST binary classifiers.

and the same training parameters $C = 1000$ and $\tau = 0.001$. Unless indicated otherwise, the kernel cache size is 256MB.

LASVM versus Sequential Minimal Optimization Baseline results were obtained by running the state-of-the-art SMO solver LIBSVM (Chang and Lin, 2001-2004). The resulting classifier accurately represents the SVM solution.

Two sets of results are reported for LASVM. The LASVM \times 1 results were obtained by performing a single epoch of online iterations: each training example was processed exactly once during a single sequential sweep over the training set. The LASVM \times 2 results were obtained by performing two epochs of online iterations.

Figures 1 and 2 show the resulting test errors and training times. LASVM \times 1 runs about three times faster than LIBSVM and yields test error rates very close to the LIBSVM results. Standard paired significance tests indicate that these small differences are not significant. LASVM \times 2 usually runs faster than LIBSVM and very closely tracks the LIBSVM test errors.

Neither the LASVM \times 1 or LASVM \times 2 experiments yield the exact SVM solution. On this data set, LASVM reaches the exact SVM solution after about five epochs. The first two epochs represent the bulk of the computing time. The remaining epochs run faster when the kernel cache is large

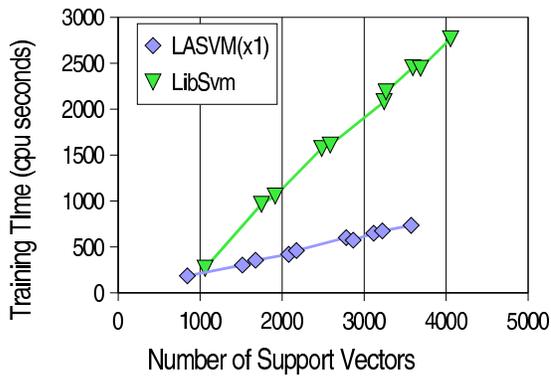


Figure 3: Training time as a function of the number of support vectors.

Algorithm	Error	Time
LIBSVM	1.36%	17400s
LASVM×1	1.42%	4950s
LASVM×2	1.36%	12210s

Figure 4: Multiclass errors and training times for the MNIST data set.

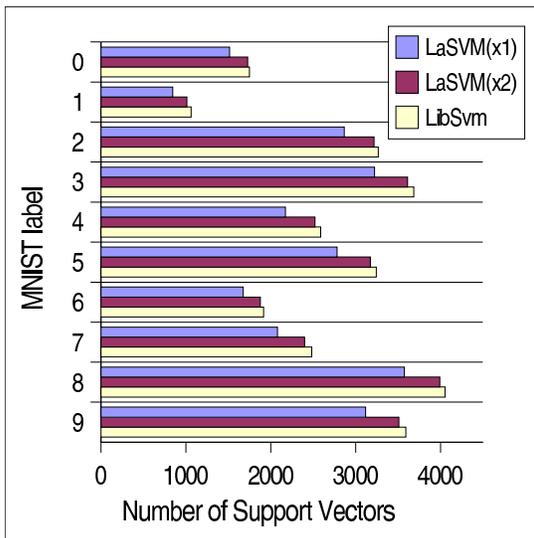


Figure 5: Compared numbers of support vectors for the ten MNIST binary classifiers.

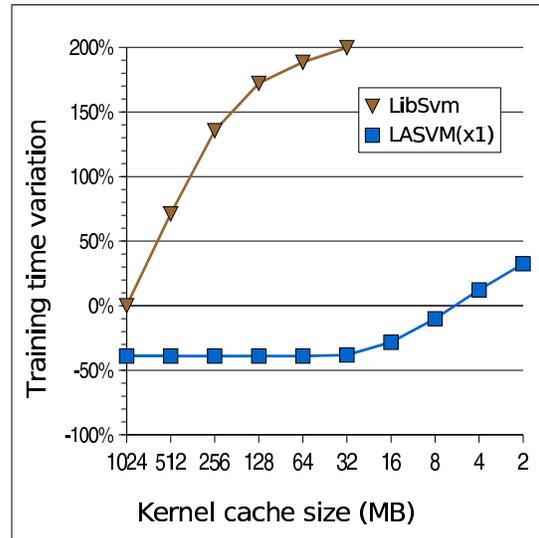


Figure 6: Training time variation as a function of the cache size. Relative changes with respect to the 1GB LIBSVM times are averaged over all ten MNIST classifiers.

enough to hold all the dot products involving support vectors. Yet the overall optimization times are not competitive with those achieved by LIBSVM.

Figure 3 shows the training time as a function of the final number of support vectors for the ten binary classification problems. Both LIBSVM and LASVM×1 show a linear dependency. The Online LASVM algorithm seems more efficient overall.

Figure 4 shows the multiclass error rates and training times obtained by combining the ten classifiers using the well known 1-versus-rest scheme (Schölkopf and Smola, 2002). $LASVM \times 1$ provides almost the same accuracy with much shorter training times. $LASVM \times 2$ reproduces the LIBSVM accuracy with slightly shorter training time.

Figure 5 shows the resulting number of support vectors. A single epoch of the Online LASVM algorithm gathers most of the support vectors of the SVM solution computed by LIBSVM. The first iterations of the Online LASVM might indeed ignore examples that later become support vectors. Performing a second epoch captures most of the missing support vectors.

LASVM versus the Averaged Perceptron The computational advantage of LASVM relies on its apparent ability to match the SVM accuracies after a single epoch. Therefore it must be compared with algorithms such as the Averaged Perceptron (Freund and Schapire, 1998) that provably match well known upper bounds on the SVM accuracies. The $AVGPERC \times 1$ results in Figures 1 and 2 were obtained after running a single epoch of the Averaged Perceptron. Although the computing times are very good, the corresponding test errors are not competitive with those achieved by either LIBSVM or LASVM. Freund and Schapire (1998) suggest that the Averaged Perceptron approaches the actual SVM accuracies after 10 to 30 epochs. Doing so no longer provides the theoretical guarantees. The $AVGPERC \times 10$ results in Figures 1 and 2 were obtained after ten epochs. Test error rates indeed approach the SVM results. The corresponding training times are no longer competitive.

Impact of the Kernel Cache Size These training times stress the importance of the kernel cache size. Figure 2 shows how the $AVGPERC \times 10$ runs much faster on problems 0, 1, and 6. This is happening because the cache is large enough to accommodate the dot products of all examples with all support vectors. Each repeated iteration of the Average Perceptron requires very few additional kernel evaluations. This is much less likely to happen when the training set size increases. Computing times then increase drastically because repeated kernel evaluations become necessary.

Figure 6 compares how the LIBSVM and $LASVM \times 1$ training times change with the kernel cache size. The vertical axis reports the relative changes with respect to LIBSVM with one gigabyte of kernel cache. These changes are averaged over the ten MNIST classifiers. The plot shows how LASVM tolerates much smaller caches. On this problem, *LASVM with a 8MB cache runs slightly faster than LIBSVM with a 1024MB cache.*

Useful orders of magnitude can be obtained by evaluating how large the kernel cache must be to avoid the systematic recomputation of dot-products. Following the notations of Section 2.1, let n be the number of examples, S be the number of support vectors, and $R \leq S$ the number of support vectors such that $0 < |\alpha_i| < C$.

- In the case of LIBSVM, the cache must accommodate about nR terms: the examples selected for the SMO iterations are usually chosen among the R free support vectors. Each SMO iteration needs n distinct dot-products for each selected example.
- To perform a *single* LASVM epoch, the cache must only accommodate about SR terms: since the examples are visited only once, the dot-products computed by a PROCESS operation can only be reutilized by subsequent REPROCESS operations. The examples selected by REPROCESS are usually chosen among the R free support vectors; for each selected example, REPROCESS needs one distinct dot-product per support vector in set S .

- To perform *multiple* LASVM epochs, the cache must accommodate about nS terms: the dot-products computed by processing a particular example are reused when processing the same example again in subsequent epochs. This also applies to multiple Averaged Perceptron epochs.

An efficient single epoch learning algorithm is therefore very desirable when one expects S to be much smaller than n . Unfortunately, this may not be the case when the data set is noisy. Section 3.4 presents results obtained in such less favorable conditions. Section 4 then proposes an active learning method to contain the growth of the number of support vectors, and recover the full benefits of the online approach.

3.4 Multiple Data Set Experiments

Further experiments were carried out with a collection of standard data sets representing diverse noise conditions, training set sizes, and input dimensionality. Figure 7 presents these data sets and the parameters used for the experiments.

Kernel computation times for these data sets are extremely fast. The data either has low dimensionality or can be represented with sparse vectors. For instance, computing kernel values for two Reuters documents only involves words common to both documents (excluding stop words). The Forest experiments use a kernel implemented with hand optimized assembly code (Graf et al., 2005).

Figure 8 compares the solutions returned by $\text{LASVM} \times 1$ and LIBSVM. The $\text{LASVM} \times 1$ experiments call the kernel function much less often, but do not always run faster. The fast kernel computation times expose the relative weakness of our kernel cache implementation. The $\text{LASVM} \times 1$ accuracies are very close to the LIBSVM accuracies. The number of support vectors is always slightly smaller.

$\text{LASVM} \times 1$ essentially achieves consistent results over very diverse data sets, after performing one single epoch over the training set only. In this situation, the LASVM PROCESS function gets only once chance to take a particular example into the kernel expansion and potentially make it a support vector. The conservative strategy would be to take all examples and sort them out during the finishing step. The resulting training times would always be worse than LIBSVM's because the finishing step is itself a simplified SMO solver. Therefore LASVM online iterations are able to very quickly discard a large number of examples with a high confidence. This process is not perfect because we can see that the $\text{LASVM} \times 1$ number of support vectors are smaller than LIBSVM's. Some good support vectors are discarded erroneously.

Figure 9 reports the relative variations of the test error, number of support vectors, and training time measured before and after the finishing step. The online iterations pretty much select the right support vectors on clean data sets such as "Waveform", "Reuters" or "USPS", and the finishing step does very little. On the other problems the online iterations keep much more examples as potential support vectors. The finishing step significantly improves the accuracy on noisy data sets such as "Banana", "Adult" or "USPS+N", and drastically increases the computation time on data sets with complicated decision boundaries such as "Banana" or "Forest".

	Train Size	Test Size	γ	C	Cache	τ	Notes
Waveform ¹	4000	1000	0.05	1	40M	0.001	Artificial data, 21 dims.
Banana ¹	4000	1300	0.5	316	40M	0.001	Artificial data, 2 dims.
Reuters ²	7700	3299	1	1	40M	0.001	Topic "moneyfx" vs. rest.
USPS ³	7329	2000	2	1000	40M	0.001	Class "0" vs. rest.
USPS+N ³	7329	2000	2	10	40M	0.001	10% training label noise.
Adult ³	32562	16282	0.005	100	40M	0.001	As in (Platt, 1999).
Forest ³ (100k)	100000	50000	1	3	512M	0.001	As in (Collobert et al., 2002).
Forest ³ (521k)	521012	50000	1	3	1250M	0.01	As in (Collobert et al., 2002).

¹ <http://mlg.anu.edu.au/~raetsch/data/index.html>

² <http://www.daviddlewis.com/resources/testcollections/reuters21578>

³ <ftp://ftp.ics.uci.edu/pub/machine-learning-databases>

Figure 7: Data Sets discussed in Section 3.4.

Data Set	LIBSVM				LASVM×1			
	Error	SV	KCalc	Time	Error	SV	KCalc	Time
Waveform	8.82%	1006	4.2M	3.2s	8.68%	948	2.2M	2.7s
Banana	9.96%	873	6.8M	9.9s	9.98%	869	6.7M	10.0s
Reuters	2.76%	1493	11.8M	24s	2.76%	1504	9.2M	31.4s
USPS	0.41%	236	1.97M	13.5s	0.43%	201	1.08M	15.9s
USPS+N	0.41%	2750	63M	305s	0.53%	2572	20M	178s
Adult	14.90%	11327	1760M	1079s	14.94%	11268	626M	809s
Forest (100k)	8.03%	43251	27569M	14598s	8.15%	41750	18939M	10310s
Forest (521k)	4.84%	124782	316750M	159443s	4.83%	122064	188744M	137183s

Figure 8: Comparison of LIBSVM versus LASVM×1: Test error rates (Error), number of support vectors (SV), number of kernel calls (KCalc), and training time (Time). Bold characters indicate significant differences.

Data Set	Relative Variation		
	Error	SV	Time
Waveform	-0%	-0%	+4%
Banana	-79%	-74%	+185%
Reuters	0%	-0%	+3%
USPS	0%	-2%	+0%
USPS+N%	-67%	-33%	+7%
Adult	-13%	-19%	+80%
Forest (100k)	-1%	-24%	+248%
Forest (521k)	-2%	-24%	+84%

Figure 9: Relative variations of test error, number of support vectors and training time measured before and after the finishing step.

3.5 The Collection of Potential Support Vectors

The final step of the REPROCESS operation computes the current value of the kernel expansion bias b and the stopping criterion δ :

$$\begin{aligned} g_{\max} &= \max_{s \in \mathcal{S}} g_s \quad \text{with } \alpha_s < B_s & b &= \frac{g_{\max} + g_{\min}}{2} \\ g_{\min} &= \min_{s \in \mathcal{S}} g_s \quad \text{with } \alpha_s > A_s & \delta &= g_{\max} - g_{\min}. \end{aligned} \quad (10)$$

The quantities g_{\min} and g_{\max} can be interpreted as bounds for the decision threshold b . The quantity δ then represents an uncertainty on the decision threshold b .

The quantity δ also controls how LASVM collects potential support vectors. The definition of PROCESS and the equality (9) indicate indeed that PROCESS(k) adds the support vector x_k to the kernel expansion if and only if

$$y_k \hat{y}(x_k) < 1 + \frac{\delta}{2} - \tau. \quad (11)$$

When α is optimal, the uncertainty δ is zero, and this condition matches the Karush-Kuhn-Tucker condition for support vectors $y_k \hat{y}(x_k) \leq 1$.

Intuitively, relation (11) describes how PROCESS collects potential support vectors that are compatible with the current uncertainty level δ on the threshold b . Simultaneously, the REPROCESS operations reduce δ and discard the support vectors that are no longer compatible with this reduced uncertainty.

The online iterations of the LASVM algorithm make equal numbers of PROCESS and REPROCESS for purely heuristic reasons. Nothing guarantees that this is the optimal proportion. The results reported in Figure 9 clearly suggest to investigate this arbitrage more closely.

Variations on REPROCESS Experiments were carried out with a slightly modified LASVM algorithm: instead of performing a single REPROCESS, the modified online iterations repeatedly run REPROCESS until the uncertainty δ becomes smaller than a predefined threshold δ_{\max} .

Figure 10 reports comparative results for the ‘‘Banana’’ data set. Similar results were obtained with other data sets. The three plots report test error rates, training time, and number of support vectors as a function of δ_{\max} . These measurements were performed after one epoch of online iterations without finishing step, and after one and two epochs followed by the finishing step. The corresponding LIBSVM figures are indicated by large triangles on the right side of the plot.

Regardless of δ_{\max} , the SVM test error rate can be replicated by performing two epochs followed by a finishing step. However, this does not guarantee that the optimal SVM solution has been reached.

Large values of δ_{\max} essentially correspond to the unmodified LASVM algorithm. Small values of δ_{\max} considerably increases the computation time because each online iteration calls REPROCESS many times in order to sufficiently reduce δ . Small values of δ_{\max} also remove the LASVM ability to produce a competitive result after a single epoch followed by a finishing step. The additional optimization effort discards support vectors more aggressively. Additional epochs are necessary to recapture the support vectors that should have been kept.

There clearly is a sweet spot around $\delta_{\max} = 3$ when one epoch of online iterations alone almost match the SVM performance and also makes the finishing step very fast. This sweet spot is difficult to find in general. If δ_{\max} is a little bit too small, we must make one extra epoch. If δ_{\max} is a little

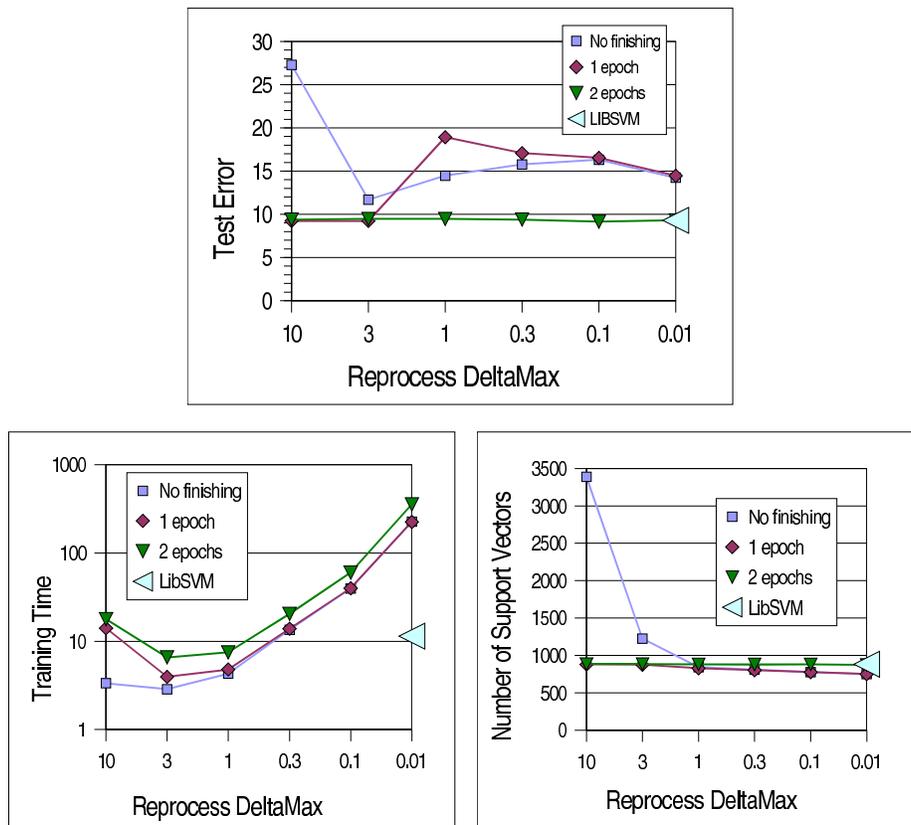


Figure 10: Impact of additional REPROCESS measured on “Banana” data set. During the LASVM online iterations, calls to REPROCESS are repeated until $\delta < \delta_{\max}$.

bit too large, the algorithm behaves like the unmodified LASVM. Short of a deeper understanding of these effects, the unmodified LASVM seems to be a robust compromise.

SimpleSVM The right side of each plot in Figure 10 corresponds to an algorithm that optimizes the coefficient of the current support vectors at each iteration. This is closely related to the SimpleSVM algorithm (Vishwanathan et al., 2003). Both LASVM and the SimpleSVM update a current kernel expansion by adding or removing one or two support vectors at each iteration. The two key differences are the numerical objective of these updates and their computational costs.

Whereas each SimpleSVM iteration seeks the optimal solution of the SVM QP problem restricted to the current set of support vectors, the LASVM online iterations merely attempt to improve the value of the dual objective function $W(\alpha)$. As a consequence, LASVM needs a finishing step and the SimpleSVM does not. On the other hand, Figure 10 suggests that seeking the optimum at each iteration discards support vectors too aggressively to reach competitive accuracies after a single epoch.

Each SimpleSVM iteration updates the current kernel expansion using rank 1 matrix updates (Cauwenberghs and Poggio, 2001) whose computational cost grows as the square of the number of support vectors. LASVM performs these updates using SMO direction searches whose cost grows

linearly with the number of examples. Rank 1 updates make good sense when one seeks the optimal coefficients. On the other hand, all the kernel values involving support vectors must be stored in memory. The LASVM direction searches are more amenable to caching strategies for kernel values.

4. Active Selection of Training Examples

The previous section presents LASVM as an Online Learning algorithm or as a Stochastic Optimization algorithm. In both cases, the LASVM online iterations pick random training examples. The current section departs from this framework and investigates more refined ways to select an informative example for each iteration.

This departure is justified in the offline setup because the complete training set is available beforehand and can be searched for informative examples. It is also justified in the online setup when the continuous stream of fresh training examples is too costly to process, either because the computational requirements are too high, or because it is impractical to label all the potential training examples.

In particular, we show that selecting informative examples yields considerable speedups. Furthermore, training example selection can be achieved without the knowledge of the training example labels. In fact, excessive reliance on the training example labels can have very detrimental effects.

4.1 Gradient Selection

The most obvious approach consists in selecting an example k such that the PROCESS operation results in a large increase of the dual objective function. This can be approximated by choosing the example which yields the τ -violating pair with the largest gradient. Depending on the class y_k , the PROCESS(k) operation considers pair (k, j) or (i, k) where i and j are the indices of the examples in \mathcal{S} with extreme gradients:

$$i = \arg \max_{s \in \mathcal{S}} g_s \text{ with } \alpha_s < B_s, \quad j = \arg \min_{s \in \mathcal{S}} g_s \text{ with } \alpha_s > A_s.$$

The corresponding gradients are $g_k - g_j$ for positive examples and $g_i - g_k$ for negative examples. Using the expression (9) of the gradients and the value of b and δ computed during the previous REPROCESS (10), we can write:

$$\begin{aligned} \text{when } y_k = +1, \quad g_k - g_j &= y_k g_k - \frac{g_i + g_j}{2} + \frac{g_i - g_j}{2} = 1 + \frac{\delta}{2} - y_k \hat{y}(x_k) \\ \text{when } y_k = -1, \quad g_i - g_k &= \frac{g_i + g_j}{2} + \frac{g_i - g_j}{2} + y_k g_k = 1 + \frac{\delta}{2} - y_k \hat{y}(x_k). \end{aligned}$$

This expression shows that the *Gradient Selection Criterion* simply suggests to pick the most misclassified example

$$k_G = \arg \min_{k \notin \mathcal{S}} y_k \hat{y}(x_k). \tag{12}$$

4.2 Active Selection

Always picking the most misclassified example is reasonable when one is very confident of the training example labels. On noisy data sets, this strategy is simply going to pick mislabelled examples or examples that sit on the wrong side of the optimal decision boundary.

When training example labels are unreliable, a conservative approach chooses the example k_A that yields the strongest minimax gradient:

$$k_A = \operatorname{argmin}_{k \notin S} \max_{y=\pm 1} y \hat{y}(x_k) = \operatorname{argmin}_{k \notin S} |\hat{y}(x_k)|. \quad (13)$$

This *Active Selection Criterion* simply chooses the example that comes closest to the current decision boundary. Such a choice yields a gradient approximatively equal to $1 + \delta/2$ regardless of the true class of the example.

Criterion (13) does not depend on the labels y_k . The resulting learning algorithm only uses the labels of examples that have been selected during the previous online iterations. This is related to the *Pool Based Active Learning* paradigm (Cohn et al., 1990).

Early active learning literature, also known as *Experiment Design* (Fedorov, 1972), contrasts the passive learner, who observes examples (x, y) , with the active learner, who constructs queries x and observes their labels y . In this setup, the active learner cannot beat the passive learner because he lacks information about the input pattern distribution (Eisenberg and Rivest, 1990). Pool-based active learning algorithms observe the pattern distribution from a vast pool of unlabelled examples. Instead of constructing queries, they incrementally select unlabelled examples x_k and obtain their labels y_k from an oracle.

Several authors (Campbell et al., 2000; Schohn and Cohn, 2000; Tong and Koller, 2000) propose incremental active learning algorithms that clearly are related to Active Selection. The initialization consists of obtaining the labels for a small random subset of examples. A SVM is trained using all the labelled examples as a training set. Then one searches the pool for the unlabelled example that comes closest to the SVM decision boundary, one obtains the label of this example, retrain the SVM and reiterates the process.

4.3 Randomized Search

Both criteria (12) and (13) suggest a search through all the training examples. This is impossible in the online setup and potentially expensive in the offline setup.

It is however possible to locate an approximate optimum by simply examining a small constant number of randomly chosen examples. The randomized search first samples M random training examples and selects the best one among these M examples. With probability $1 - \eta^M$, the value of the criterion for this example exceeds the η -quantile of the criterion for all training examples (Schölkopf and Smola, 2002, theorem 6.33) regardless of the size of the training set. In practice this means that the best among 59 random training examples has 95% chances to belong to the best 5% examples in the training set.

Randomized search has been used in the offline setup to accelerate various machine learning algorithms (Domingo and Watanabe, 2000; Vishwanathan et al., 2003; Tsang et al., 2005). In the online setup, randomized search is the only practical way to select training examples. For instance, here is a modification of the basic LASVM algorithm to select examples using the Active Selection Criterion with Randomized Search:

LASVM + Active Example Selection + Randomized Search**1) Initialization:**

Seed \mathcal{S} with a few examples of each class.
Set $\alpha \leftarrow 0$ and $g \leftarrow 0$.

2) Online Iterations:

Repeat a predefined number of times:

- Pick M random examples $s_1 \dots s_M$.
- $k_t \leftarrow \arg \min_{i=1 \dots M} |\hat{y}(x_{s_i})|$
- Run PROCESS(k_t).
- Run REPROCESS once.

3) Finishing:

Repeat REPROCESS until $\delta \leq \tau$.

Each online iteration of the above algorithm is about M times more computationally expensive than an online iteration of the basic LASVM algorithm. Indeed one must compute the kernel expansion (2) for M fresh examples instead of a single one (9). This cost can be reduced by heuristic techniques for adapting M to the current conditions. For instance, we present experimental results where one stops collecting new examples as soon as \mathcal{M} contains five examples such that $|\hat{y}(x_s)| < 1 + \delta/2$.

Finally the last two paragraphs of appendix A discuss the convergence of LASVM with example selection according to the gradient selection criterion or the active selection criterion. The gradient selection criterion always leads to a solution of the SVM problem. On the other hand, the active selection criterion only does so when one uses the sampling method. In practice this convergence occurs very slowly. The next section presents many reasons to prefer the intermediate kernel classifiers visited by this algorithm.

4.4 Example Selection for Online SVMs

This section experimentally compares the LASVM algorithm using different example selection methods. Four different algorithms are compared:

- RANDOM example selection randomly picks the next training example among those that have not yet been PROCESSED. This is equivalent to the plain LASVM algorithm discussed in Section 3.2.
- GRADIENT example selection consists in sampling 50 random training examples among those that have not yet been PROCESSED. The sampled example with the smallest $y_k \hat{y}(x_k)$ is then selected.
- ACTIVE example selection consists in sampling 50 random training examples among those that have not yet been PROCESSED. The sampled example with the smallest $|\hat{y}(x_k)|$ is then selected.
- AUTOACTIVE example selection attempts to adaptively select the sampling size. Sampling stops as soon as 5 examples are within distance $1 + \delta/2$ of the decision boundary. The maximum sample size is 100 examples. The sampled example with the smallest $|\hat{y}(x_k)|$ is then selected.

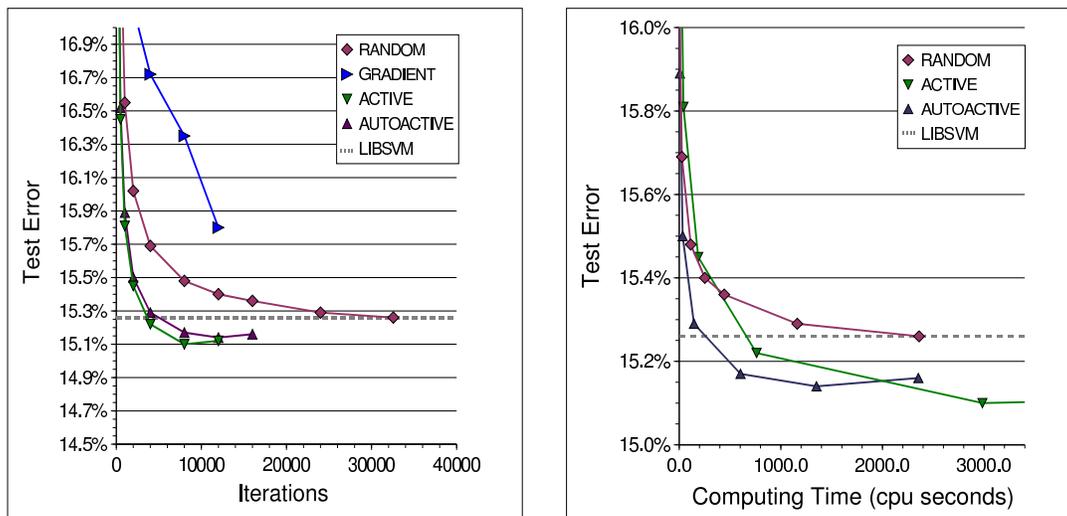


Figure 11: Comparing example selection criteria on the Adult data set. Measurements were performed on 65 runs using randomly selected training sets. The graphs show the error measured on the remaining testing examples as a function of the number of iterations and the computing time. The dashed line represents the LIBSVM test error under the same conditions.

Adult Data Set We first report experiments performed on the “Adult” data set. This data set provides a good indication of the relative performance of the Gradient and Active selection criteria under noisy conditions.

Reliable results were obtained by averaging experimental results measured for 65 random splits of the full data set into training and test sets. Paired tests indicate that test error differences of 0.25% on a single run are statistically significant at the 95% level. We conservatively estimate that average error differences of 0.05% are meaningful.

Figure 11 reports the average error rate measured on the test set as a function of the number of online iterations (left plot) and of the average computing time (right plot). Regardless of the training example selection method, all reported results were measured after performing the LASVM finishing step. More specifically, we run a predefined number of online iterations, save the LASVM state, perform the finishing step, measure error rates and number of support vectors, and restore the saved LASVM state before proceeding with more online iterations. Computing time includes the duration of the online iterations and the duration of the finishing step.

The GRADIENT example selection criterion performs very poorly on this noisy data set. A detailed analysis shows that most of the selected examples become support vectors with coefficient reaching the upper bound C . The ACTIVE and AUTOACTIVE criteria both reach smaller test error rates than those achieved by the SVM solution computed by LIBSVM. The error rates then seem to increase towards the error rate of the SVM solution (left plot). We believe indeed that continued iterations of the algorithm eventually yield the SVM solution.

Figure 12 relates error rates and numbers of support vectors. The RANDOM LASVM algorithm performs as expected: a single pass over all training examples replicates the accuracy and the num-

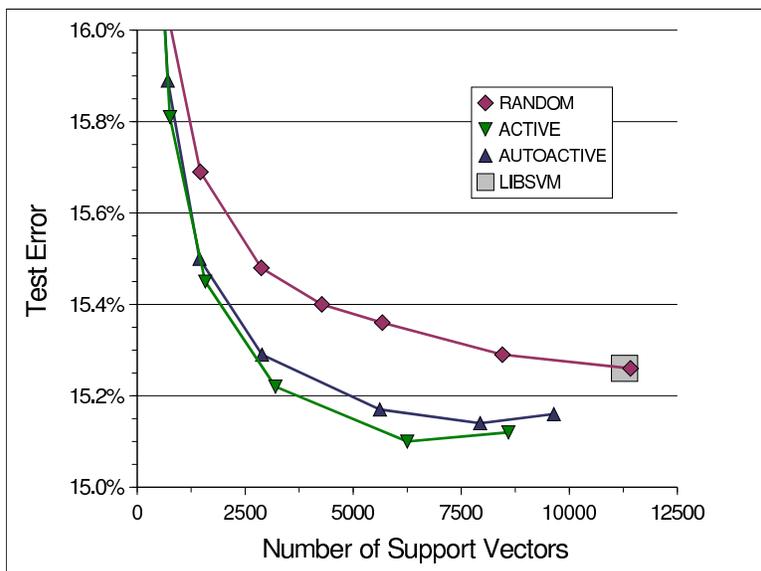


Figure 12: Comparing example selection criteria on the Adult data set. Test error as a function of the number of support vectors.

ber of support vectors of the LIBSVM solution. Both the ACTIVE and AUTOACTIVE criteria yield kernel classifiers with the same accuracy and much less support vectors. For instance, the AUTOACTIVE LASVM algorithm reaches the accuracy of the LIBSVM solution using 2500 support vectors instead of 11278. Figure 11 (right plot) shows that this result is achieved after 150 seconds only. This is about one fifteenth of the time needed to perform a full RANDOM LASVM epoch.²

Both the ACTIVE LASVM and AUTOACTIVE LASVM algorithms exceed the LIBSVM accuracy after a few iterations only. This is surprising because these algorithms only use the training labels of the few selected examples. They both outperform the LIBSVM solution by using only a small subset of the available training labels.

MNIST Data Set The comparatively clean MNIST data set provides a good opportunity to verify the behavior of the various example selection criteria on a problem with a much lower error rate.

Figure 13 compares the performance of the RANDOM, GRADIENT and ACTIVE criteria on the classification of digit “8” versus all other digits. The curves are averaged on 5 runs using different random seeds. All runs use the standard MNIST training and test sets. Both the GRADIENT and ACTIVE criteria perform similarly on this relatively clean data set. They require about as much computing time as RANDOM example selection to achieve a similar test error.

Adding ten percent label noise on the MNIST training data provides additional insight regarding the relation between noisy data and example selection criteria. Label noise was not applied to the testing set because the resulting measurement can be readily compared to test errors achieved by training SVMs without label noise. The expected test errors under similar label noise conditions can be derived from the test errors measured without label noise. Figure 14 shows the test errors achieved when 10% label noise is added to the training examples. The GRADIENT selection cri-

2. The timing results reported in Figure 8 were measured on a faster computer.

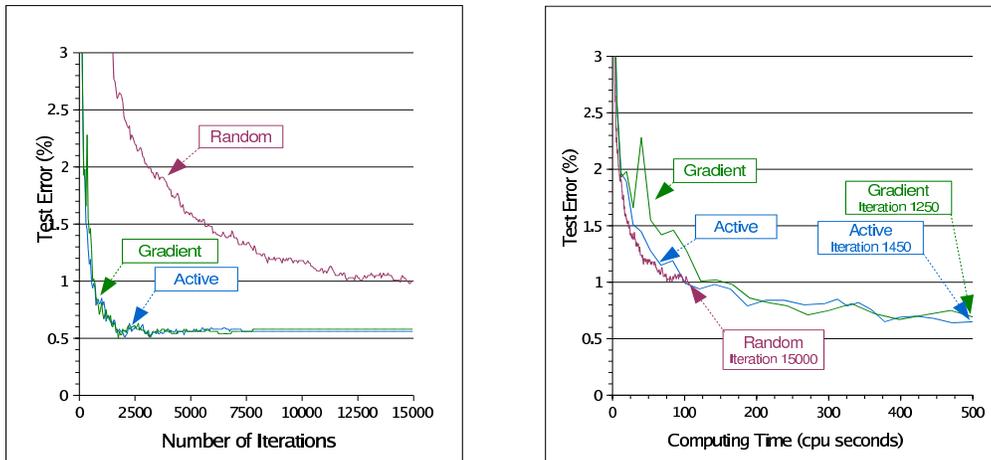


Figure 13: Comparing example selection criteria on the MNIST data set, recognizing digit “8” against all other classes. Gradient selection and Active selection perform similarly on this relatively noiseless task.

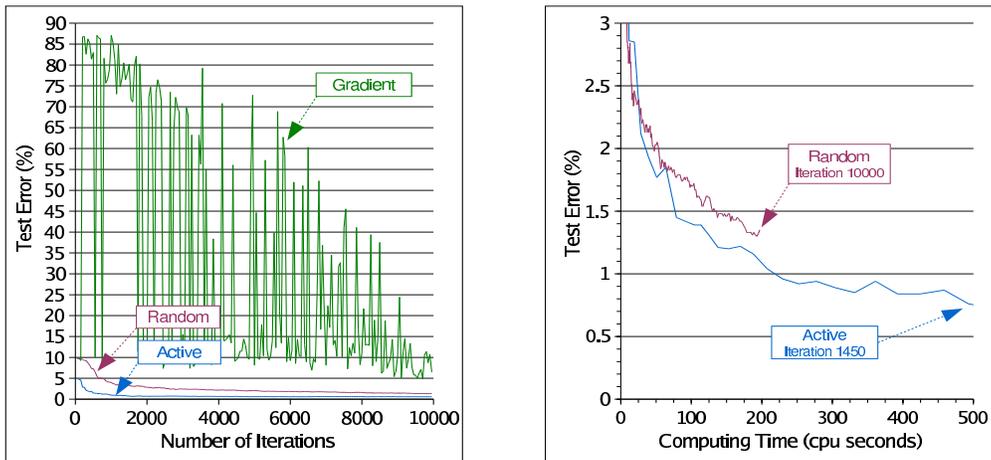


Figure 14: Comparing example selection criteria on the MNIST data set with 10% label noise on the training examples.

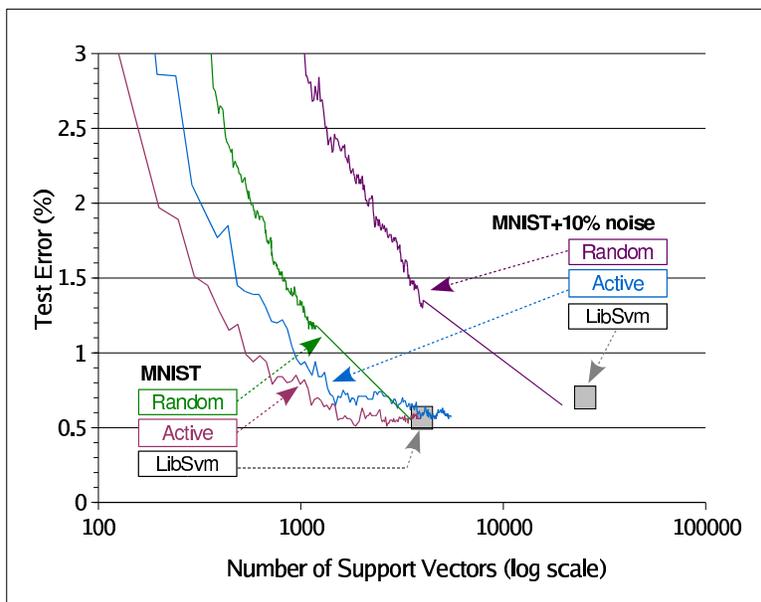


Figure 15: Comparing example selection criteria on the MNIST data set. Active example selection is insensitive to the artificial label noise.

terion causes a very chaotic convergence because it keeps selecting mislabelled training examples. The ACTIVE selection criterion is obviously undisturbed by the label noise.

Figure 15 summarizes error rates and number of support vectors for all noise conditions. In the presence of label noise on the training data, LIBSVM yields a slightly higher test error rate, and a much larger number of support vectors. The RANDOM LASVM algorithm replicates the LIBSVM results after one epoch. Regardless of the noise conditions, the ACTIVE LASVM algorithm reaches the accuracy and the number of support vectors of the LIBSVM solution obtained with clean training data. Although we have not been able to observe it on this data set, we expect that, after a very long time, the ACTIVE curve for the noisy training set converges to the accuracy and the number of support vectors achieved of the LIBSVM solution obtained for the noisy training data.

4.5 Online SVMs for Active Learning

The ACTIVE LASVM algorithm implements two dramatic speedups with respect to existing active learning algorithms such as (Campbell et al., 2000; Schohn and Cohn, 2000; Tong and Koller, 2000). First it chooses a query by sampling a small number of random examples instead of scanning all unlabelled examples. Second, it uses a single LASVM iteration after each query instead of fully retraining the SVM.

Figure 16 reports experiments performed on the Reuters and USPS data sets presented in table 7. The RETRAIN ACTIVE 50 and RETRAIN ACTIVE ALL select a query from 50 or all unlabeled examples respectively, and then retrain the SVM. The SVM solver was initialized with the solution from the previous iteration. The LASVM ACTIVE 50 and LASVM ACTIVE ALL do not retrain the SVM, but instead make a single LASVM iteration for each new labeled example.

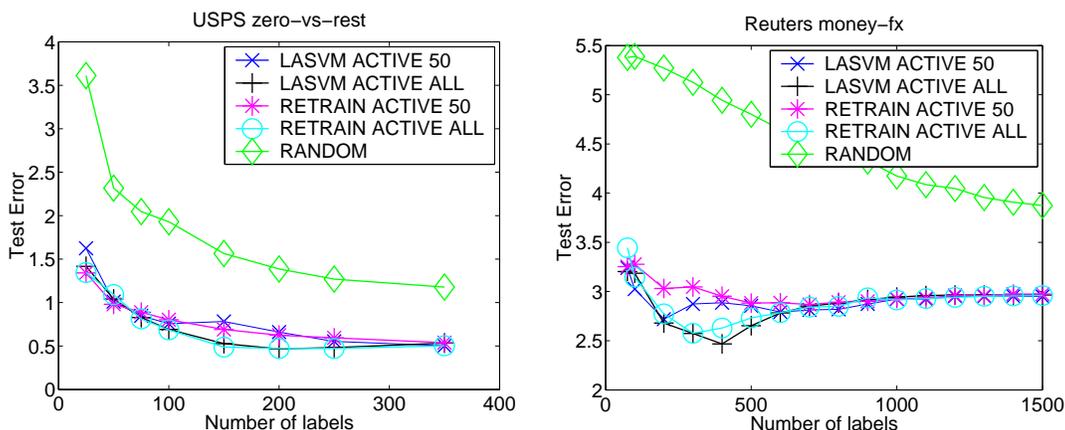


Figure 16: Comparing active learning methods on the USPS and Reuters data sets. Results are averaged on 10 random choices of training and test sets. Using LASVM iterations instead of retraining causes no loss of accuracy. Sampling $M = 50$ examples instead of searching all examples only causes a minor loss of accuracy when the number of labeled examples is very small.

All the active learning methods performed approximately the same, and were superior to random selection. Using LASVM iterations instead of retraining causes no loss of accuracy. Sampling $M = 50$ examples instead of searching all examples only causes a minor loss of accuracy when the number of labeled examples is very small. Yet the speedups are very significant: for 500 queried labels on the Reuters data set, the RETRAIN ACTIVE ALL, LASVM ACTIVE ALL, and LASVM ACTIVE 50 algorithms took 917 seconds, 99 seconds, and 9.6 seconds respectively.

5. Discussion

This work started because we observed that the data set sizes are quickly outgrowing the computing power of our calculators. One possible avenue consists of harnessing the computing power of multiple computers (Graf et al., 2005). Instead we propose learning algorithms that remain closely related to SVMs but require less computational resources. This section discusses their practical and theoretical implications.

5.1 Practical Significance

When we have access to an abundant source of training examples, the simple way to reduce the complexity of a learning algorithm consists of picking a random subset of training examples and running a regular training algorithm on this subset. Unfortunately this approach renounces the more accurate models that the large training set could afford. This is why we say, by reference to statistical efficiency, that an *efficient* learning algorithm should at least pay a brief look at every training example.

The online LASVM algorithm is very attractive because it matches the performance of a SVM trained on all the examples. More importantly, it achieves this performance after a single epoch,

faster than a SVM (figure 2) and using much less memory than a SVM (figure 6). This is very important in practice because modern data storage devices are most effective when the data is accessed sequentially.

Active Selection of the LASVM training examples brings two additional benefits for practical applications. It achieves equivalent performances with significantly less support vectors, further reducing the required time and memory. It also offers an obvious opportunity to parallelize the search for informative examples.

5.2 Informative Examples and Support Vectors

By suggesting that all examples should not be given equal attention, we first state that all training examples are not equally informative. This question has been asked and answered in various contexts (Fedorov, 1972; Cohn et al., 1990; MacKay, 1992). We also ask whether these differences can be exploited to reduce the computational requirements of learning algorithms. Our work answers this question by proposing algorithms that exploit these differences and achieve very competitive performances.

Kernel classifiers in general distinguish the few training examples named support vectors. Kernel classifier algorithms usually maintain an active set of potential support vectors and work by iterations. Their computing requirements are readily associated with the training examples that belong to the active set. Adding a training example to the active set increases the computing time associated with each subsequent iteration because they will require additional kernel computations involving this new support vector. Removing a training example from the active set reduces the cost of each subsequent iteration. However it is unclear how such changes affect the number of subsequent iterations needed to reach a satisfactory performance level.

Online kernel algorithms, such as the kernel perceptrons usually produce different classifiers when given different sequences of training examples. Section 3 proposes an online kernel algorithm that converges to the SVM solution after many epochs. The final set of support vectors is intrinsically defined by the SVM QP problem, regardless of the path followed by the online learning process. Intrinsic support vectors provide a benchmark to evaluate the impact of changes in the active set of current support vectors. Augmenting the active set with an example that is not an intrinsic support vector moderately increases the cost of each iteration without clear benefits. Discarding an example that is an intrinsic support vector incurs a much higher cost. Additional iterations will be necessary to recapture the missing support vector. Empirical evidence is presented in Section 3.5.

Nothing guarantees however that the most informative examples are the support vectors of the SVM solution. Bakır et al. (2005) interpret Steinwart's theorem (Steinwart, 2004) as an indication that the number of SVM support vectors is asymptotically driven by the examples located on the wrong side of the optimal decision boundary. Although such outliers might play a useful role in the construction of a decision boundary, it seems unwise to give them the bulk of the available computing time. Section 4 adds explicit example selection criteria to LASVM. The Gradient Selection Criterion selects the example most likely to cause a large increase of the SVM objective function. Experiments show that it prefers outliers over honest examples. The Active Selection Criterion bypasses the problem by choosing examples without regard to their labels. Experiments show that it leads to competitive test error rates after a shorter time, with less support vectors, and using only the labels of a small fraction of the examples.

5.3 Theoretical Questions

The appendix provides a comprehensive analysis of the convergence of the algorithms discussed in this contribution. Such convergence results are useful but limited in scope. This section underlines some aspects of this work that would vastly benefit from a deeper theoretical understanding.

- Empirical evidence suggests that a single epoch of the LASVM algorithm yields misclassification rates comparable with a SVM. We also know that LASVM exactly reaches the SVM solution after a sufficient number of epochs. Can we theoretically estimate the expected difference between the first epoch test error and the many epoch test error? Such results exist for well designed online learning algorithms based on stochastic gradient descent (Murata and Amari, 1999; Bottou and LeCun, 2005). Unfortunately these results do not directly apply to kernel classifiers. A better understanding would certainly suggest improved algorithms.
- Test error rates are sometimes improved by active example selection. In fact this effect has already been observed in the active learning setups (Schohn and Cohn, 2000). This small improvement is difficult to exploit in practice because it requires very sensitive early stopping criteria. Yet it demands an explanation because it seems that one gets a better performance by using less information. There are three potential explanations: (i) active selection works well on unbalanced data sets because it tends to pick equal number of examples of each class (Schohn and Cohn, 2000), (ii) active selection improves the SVM loss function because it discards distant outliers, (iii) active selection leads to more sparse kernel expansions with better generalization abilities (Cesa-Bianchi et al., 2005). These three explanations may be related.
- We know that the number of SVM support vectors scales linearly with the number of examples (Steinwart, 2004). Empirical evidence suggests that active example selection yields transitory kernel classifiers that achieve low error rates with much less support vectors. What is the scaling law for this new number of support vectors?
- What is the minimal computational cost for learning n independent examples and achieving “optimal” test error rates? The answer depends of course of how we define these “optimal” test error rates. This cost intuitively scales at least linearly with n because one must pay a look at each example to fully exploit them. The present work suggest that this cost might be smaller than n times the reduced number of support vectors achievable with the active learning technique. This range is consistent with previous work showing that stochastic gradient algorithms can train a fixed capacity model in linear time (Bottou and LeCun, 2005). Learning seems to be much easier than computing the optimum of the empirical loss.

5.4 Future Directions

Progress can also be achieved along less arduous directions.

- Section 3.5 suggests that better convergence speed could be attained by cleverly modulating the number of calls to REPROCESS during the online iterations. Simple heuristics might go a long way.

- Section 4.3 suggests a heuristic to adapt the sampling size for the randomized search of informative training examples. This AUTOACTIVE heuristic performs very well and deserves further investigation.
- Sometimes one can generate a very large number of training examples by exploiting known invariances. Active example selection can drive the generation of examples. This idea was suggested in (Loosli et al., 2004) for the SimpleSVM.

6. Conclusion

This work explores various ways to speedup kernel classifiers by asking which examples deserve more computing time. We have proposed a novel online algorithm that converges to the SVM solution. LASVM reliably reaches competitive accuracies after performing a single pass over the training examples, outspeeding state-of-the-art SVM solvers. We have then shown how active example selection can yield faster training, higher accuracies and simpler models using only a fraction of the training examples labels.

Acknowledgments

Part of this work was funded by NSF grant CCR-0325463. We also thank Eric Cosatto, Hans-Peter Graf, C. Lee Giles and Vladimir Vapnik for their advice and support, Ronan Collobert and Chih-Jen Lin for thoroughly checking the mathematical appendix, and Sathiya Keerthi for pointing out reference (Takahashi and Nishi, 2003).

Appendix A. Convex Programming with Witness Families

This appendix presents theoretical elements about convex programming algorithms that rely on successive direction searches. Results are presented for the case where directions are selected from a well chosen finite pool, like SMO (Platt, 1999), and for the stochastic algorithms, like the online and active SVM discussed in the body of this contribution.

Consider a compact convex subset \mathcal{F} of \mathbb{R}^n and a concave function f defined on \mathcal{F} . We assume that f is twice differentiable with continuous derivatives. This appendix discusses the maximization of function f over set F :

$$\max_{x \in \mathcal{F}} f(x). \quad (14)$$

This discussion starts with some results about feasible directions. Then it introduces the notion of witness family of directions which leads to a more compact characterization of the optimum. Finally it presents maximization algorithms and establishes their convergence to approximate solutions

A.1 Feasible Directions

Notations Given a point $x \in \mathcal{F}$ and a direction $u \in \mathbb{R}_*^n = \mathbb{R}^n$, let

$$\begin{aligned} \phi(x, u) &= \max\{\lambda \geq 0 \mid x + \lambda u \in \mathcal{F}\} \\ f^*(x, u) &= \max\{f(x + \lambda u), x + \lambda u \in \mathcal{F}\}. \end{aligned}$$

In particular we write $\phi(x, 0) = \infty$ and $f^*(x, 0) = f(x)$.

Definition 1 *The cone of feasible directions in $x \in \mathcal{F}$ is the set*

$$\mathcal{D}_x = \{u \in \mathbb{R}^n \mid \phi(x, u) > 0\}.$$

All the points $x + \lambda u$, $0 \leq \lambda \leq \phi(x, u)$ belong to \mathcal{F} because \mathcal{F} is convex. Intuitively, a direction $u \neq 0$ is feasible in x when we can start from x and make a little movement along direction u without leaving the convex set \mathcal{F} .

Proposition 2 *Given $x \in \mathcal{F}$ and $u \in \mathbb{R}^n$,*

$$f^*(x, u) > f(x) \iff \begin{cases} u' \nabla f(x) > 0 \\ u \in \mathcal{D}_x. \end{cases}$$

Proof Assume $f^*(x, u) > f(x)$. Direction $u \neq 0$ is feasible because the maximum $f^*(x, u)$ is reached for some $0 < \lambda^* \leq \phi(x, u)$. Let $v \in [0, 1]$. Since set \mathcal{F} is convex, $x + v\lambda^*u \in \mathcal{F}$. Since function f is concave, $f(x + v\lambda^*u) \geq (1 - v)f(x) + vf^*(x, u)$. Writing a first order expansion when $v \rightarrow 0$ yields $\lambda^*u' \nabla f(x) \geq f^*(x, u) - f(x) > 0$. Conversely, assume $u' \nabla f(x) > 0$ and $u \neq 0$ is a feasible direction. Recall $f(x + \lambda u) = f(x) + \lambda u' \nabla f(x) + o(\lambda)$. Therefore we can choose $0 < \lambda_0 \leq \phi(x, u)$ such that $f(x + \lambda_0 u) > f(x) + \lambda_0 u' \nabla f(x) / 2$. Therefore $f^*(x, u) \geq f(x + \lambda_0 u) > f(x)$. ■

Theorem 3 (Zoutendijk (1960) page 22) *The following assertions are equivalent:*

- i) x is a solution of problem (14).
- ii) $\forall u \in \mathbb{R}^n \quad f^*(x, u) \leq f(x)$.
- iii) $\forall u \in \mathcal{D}_x \quad u' \nabla f(x) \leq 0$.

Proof The equivalence between assertions (ii) and (iii) results from proposition 2. Assume assertion (i) is true. Assertion (ii) is necessarily true because $f^*(x, u) \leq \max_{\mathcal{F}} f = f(x)$. Conversely, assume assertion (i) is false. Then there is $y \in \mathcal{F}$ such that $f(y) > f(x)$. Therefore $f^*(x, y - x) > f(x)$ and assertion (ii) is false. ■

A.2 Witness Families

We now seek to improve this theorem. Instead of considering all feasible directions in \mathbb{R}^n , we wish to only consider the feasible directions from a smaller set \mathcal{U} .

Proposition 4 *Let $x \in \mathcal{F}$ and $v_1 \dots v_k \in \mathcal{D}_x$ be feasible directions. Every positive linear combination of $v_1 \dots v_k$ (i.e. a linear combination with positive coefficients) is a feasible direction.*

Proof Let u be a positive linear combination of the v_i . Since the v_i are feasible directions there are $y_i = x + \lambda_i v_i \in \mathcal{F}$, and u can be written as $\sum_i \gamma_i (y_i - x)$ with $\gamma_i \geq 0$. Direction u is feasible because the convex \mathcal{F} contains $(\sum \gamma_i y_i) / (\sum \gamma_i) = x + (1 / \sum \gamma_i) u$. ■

Definition 5 A set of directions $\mathcal{U} \subset \mathbb{R}_*^n$ is a “witness family for \mathcal{F} ” when, for any point $x \in \mathcal{F}$, any feasible direction $u \in \mathcal{D}_x$ can be expressed as a positive linear combination of a finite number of feasible directions $v_j \in \mathcal{U} \cap \mathcal{D}_x$.

This definition directly leads to an improved characterization of the optima.

Theorem 6 Let \mathcal{U} be a witness family for convex set \mathcal{F} .

The following assertions are equivalent:

- i) x is a solution of problem (14).
- ii) $\forall u \in \mathcal{U} \quad f^*(x, u) \leq f(x)$.
- iii) $\forall u \in \mathcal{U} \cap \mathcal{D}_x \quad u' \nabla f(x) \leq 0$.

Proof The equivalence between assertions (ii) and (iii) results from proposition 2. Assume assertion (i) is true. Theorem 3 implies that assertion (ii) is true as well. Conversely, assume assertion (i) is false. Theorem 3 implies that there is a feasible direction $u \in \mathbb{R}^n$ on point x such that $u' \nabla f(x) > 0$. Since \mathcal{U} is a witness family, there are positive coefficients $\gamma_1 \dots \gamma_k$ and feasible directions $v_1, \dots, v_k \in \mathcal{U} \cap \mathcal{D}_x$ such that $u = \sum \gamma_i v_i$. We have then $\sum \gamma_j v_j' \nabla f(x) > 0$. Since all coefficients γ_j are positive, there is at least one term j_0 such that $v_{j_0}' \nabla f(x) > 0$. Assertion (iii) is therefore false. ■

The following proposition provides an example of witness family for the convex domain \mathcal{F}_s that appears in the SVM QP problem (5).

Proposition 7 Let $(e_1 \dots e_n)$ be the canonical basis of \mathbb{R}^n . Set $\mathcal{U}_s = \{e_i - e_j, i \neq j\}$ is a witness family for convex set \mathcal{F}_s defined by the constraints

$$x \in \mathcal{F}_s \iff \begin{cases} \forall i \quad A_i \leq x_i \leq B_i \\ \sum_i x_i = 0. \end{cases}$$

Proof Let $u \in \mathbb{R}_*^n$ be a feasible direction in $x \in \mathcal{F}_s$. Since u is a feasible direction, there is $\lambda > 0$ such that $y = x + \lambda u \in \mathcal{F}_s$. Consider the subset $\mathcal{B} \subset \mathcal{F}_s$ defined by the constraints

$$z \in \mathcal{B} \iff \begin{cases} \forall i, A_i \leq \min(x_i, y_i) \leq z_i \leq \max(x_i, y_i) \leq B_i \\ \sum_i z_i = 0. \end{cases}$$

Let us recursively define a sequence of points $z(j) \in \mathcal{B}$. We start with $z(0) = x \in \mathcal{B}$. For each $t \geq 0$, we define two sets of coordinate indices $I_t^+ = \{i | z_i(t) < y_i\}$ and $I_t^- = \{j | z_j(t) > y_j\}$. The recursion stops if either set is empty. Otherwise, we choose $i \in I_t^+$ and $j \in I_t^-$ and define $z(t+1) = z(t) + \gamma(t) v(t) \in \mathcal{B}$ with $v(t) = e_i - e_j \in \mathcal{U}_s$ and $\gamma(t) = \min(y_i - z_i(t), z_j(t) - y_j) > 0$. Intuitively, we move towards y along direction $v(t)$ until we hit the boundaries of set \mathcal{B} .

Each iteration removes at least one of the indices i or j from sets I_t^+ and I_t^- . Eventually one of these sets gets empty and the recursion stops after a finite number k of iterations. The other set is also empty because

$$\sum_{i \in I_k^+} |y_i - z_i(k)| - \sum_{i \in I_k^-} |y_i - z_i(k)| = \sum_{i=1}^n y_i - z_i(k) = \sum_{i=1}^n y_i - \sum_{i=1}^n z_i(k) = 0.$$

Therefore $z(k) = y$ and $\lambda u = y - x = \sum_t \gamma(t) v(t)$. Moreover the $v(t)$ are feasible directions on x because $v(t) = e_i - e_j$ with $i \in I_t^+ \subset I_0^+$ and $j \in I_t^- \subset I_0^-$. ■

Assertion (iii) in Theorem 6 then yields the following necessary and sufficient optimality criterion for the SVM QP problem (5):

$$\forall (i, j) \in \{1 \dots n\}^2 \quad x_i < B_i \text{ and } x_j > A_j \Rightarrow \frac{\partial f}{\partial x_i}(x) - \frac{\partial f}{\partial x_j}(x) \leq 0.$$

Different constraint sets call for different choices of witness family. For instance, it is sometimes useful to disregard the equality constraint in the SVM polytope \mathcal{F}_s . Along the lines of proposition 7, it is quite easy to prove that $\{\pm e_i, i = 1 \dots n\}$ is a witness family. Theorem 6 then yields an adequate optimality criterion.

A.3 Finite Witness Families

This section deals with *finite witness families*. Theorem 9 shows that \mathcal{F} is necessarily a convex polytope, that is a bounded set defined by a finite number of linear of linear equality and inequality constraints (Schrijver, 1986).

Proposition 8 *Let $C_x = \{x + u, u \in \mathcal{D}_x\}$ for $x \in \mathcal{F}$. Then $\mathcal{F} = \bigcap_{x \in \mathcal{F}} C_x$.*

Proof We first show that $\mathcal{F} \subset \bigcap_{x \in \mathcal{F}} C_x$. Indeed $\mathcal{F} \subset C_x$ for all x because every point $z \in \mathcal{F}$ defines a feasible direction $z - x \in \mathcal{D}_x$.

Conversely, Let $z \in \bigcap_{x \in \mathcal{F}} C_x$ and assume that z does not belong to \mathcal{F} . Let \hat{z} be the projection of z on \mathcal{F} . We know that $z \in C_{\hat{z}}$ because $z \in \bigcap_{x \in \mathcal{F}} C_x$. Therefore $z - \hat{z}$ is a feasible direction in \hat{z} . Choose $0 < \lambda < \phi(\hat{z}, z - \hat{z})$. We know that $\lambda < 1$ because z does not belong to \mathcal{F} . But then $\hat{z} + \lambda(z - \hat{z}) \in \mathcal{F}$ is closer to z than \hat{z} . This contradicts the definition of the projection \hat{z} . ■

Theorem 9 *Let \mathcal{F} be a bounded convex set.*

If there is a finite witness family for \mathcal{F} , then \mathcal{F} is a convex polytope.³

Proof Consider a point $x \in \mathcal{F}$ and let $\{v_1 \dots v_k\} = \mathcal{U} \cap \mathcal{D}_x$. Proposition 4 and definition 5 imply that \mathcal{D}_x is the polyhedral cone $\{z = \sum \gamma_i v_i, \gamma_i \geq 0\}$ and can be represented (Schrijver, 1986) by a finite number of linear equality and inequality constraints of the form $nz \leq 0$ where the directions n are unit vectors. Let \mathcal{K}_x be the set of these unit vectors. Equality constraints arise when the set \mathcal{K}_x contains both n and $-n$. Each set \mathcal{K}_x depends only on the subset $\{v_1 \dots v_k\} = \mathcal{U} \cap \mathcal{D}_x$ of feasible witness directions in x . Since the finite set \mathcal{U} contains only a finite number of potential subsets, there is only a finite number of distinct sets \mathcal{K}_x .

Each set C_x is therefore represented by the constraints $nz \leq nx$ for $n \in \mathcal{K}_x$. The intersection $\mathcal{F} = \bigcap_{x \in \mathcal{F}} C_x$ is then defined by all the constraints associated with C_x for any $x \in \mathcal{F}$. These constraints involve only a finite number of unit vectors n because there is only a finite number of distinct sets \mathcal{K}_x .

Inequalities defined by the same unit vector n can be summarized by considering only the most restrictive right hand side. Therefore \mathcal{F} is described by a finite number of equality and inequality

constraints. Since \mathcal{F} is bounded, it is a polytope. ■

A convex polytope comes with useful continuity properties.

Proposition 10 *Let \mathcal{F} be a polytope, and let $u \in \mathbb{R}^n$ be fixed.*

Functions $x \mapsto \phi(x, u)$ and $x \mapsto f^(x, u)$ are uniformly continuous on \mathcal{F} .*

Proof The polytope \mathcal{F} is defined by a finite set of constraints $nx \leq b$. Let $\mathcal{K}_{\mathcal{F}}$ be the set of pairs (n, b) representing these constraints. Function $x \mapsto \phi(x, u)$ is a continuous on \mathcal{F} because we can write:

$$\phi(x, u) = \min \left\{ \frac{b - nx}{nu} \text{ for all } (n, b) \in \mathcal{K}_{\mathcal{F}} \text{ such that } nu > 0 \right\}.$$

Function $x \mapsto \phi(x, u)$ is uniformly continuous because it is continuous on the compact \mathcal{F} .

Choose $\varepsilon > 0$ and let $x, y \in \mathcal{F}$. Let the maximum $f^*(x, u)$ be reached in $x + \lambda^*u$ with $0 \leq \lambda^* \leq \phi(x, u)$. Since f is uniformly continuous on compact \mathcal{F} , there is $\eta > 0$ such that $|f(x + \lambda^*u) - f(y + \lambda'u)| < \varepsilon$ whenever $\|x - y + (\lambda^* - \lambda')u\| < \eta(1 + \|u\|)$. In particular, it is sufficient to have $\|x - y\| < \eta$ and $|\lambda^* - \lambda'| < \eta$. Since ϕ is uniformly continuous, there is $\tau > 0$ such that $|\phi(y, u) - \phi(x, u)| < \eta$ whenever $\|x - y\| < \tau$. We can then select $0 \leq \lambda' \leq \phi(y, u)$ such that $|\lambda^* - \lambda'| < \eta$. Therefore, when $\|x - y\| < \min(\eta, \tau)$, $f^*(x, u) = f(x + \lambda^*u) \leq f(y + \lambda'u) + \varepsilon \leq f^*(y, u) + \varepsilon$.

By reversing the roles of x and y in the above argument, we can similarly establish that $f^*(y, u) \leq f^*(x, u) + \varepsilon$ when $\|x - y\| \leq \min(\eta, \tau)$. Function $x \mapsto f^*(x, u)$ is therefore uniformly continuous on \mathcal{F} . ■

A.4 Stochastic Witness Direction Search

Each iteration of the following algorithm randomly chooses a feasible witness direction and performs an optimization along this direction. The successive search directions u_t are randomly selected (step 2a) according to some distribution P_t defined on \mathcal{U} . Distribution P_t possibly depends on values observed before time t .

Stochastic Witness Direction Search (WDS)

- 1) Find an initial feasible point $x_0 \in \mathcal{F}$.
- 2) For each $t = 1, 2, \dots$,
 - 2a) Draw a direction $u_t \in \mathcal{U}$ from a distribution P_t
 - 2b) If $u \in \mathcal{D}_{x_{t-1}}$ and $u_t' \nabla f(x_{t-1}) > 0$,

$$x_t \leftarrow \operatorname{argmax} f(x) \text{ under } x \in \{x_{t-1} + \lambda u_t \in \mathcal{F}, \lambda \geq 0\}$$
 otherwise

$$x_t \leftarrow x_{t-1}.$$

Clearly the Stochastic WDS algorithm does not work if the distributions P_t always give probability zero to important directions. On the other hand, convergence is easily established if all feasible directions can be drawn with non zero minimal probability at any time.

3. We believe that the converse of Theorem 9 is also true.

Theorem 11 *Let f be a concave function defined on a compact convex set \mathcal{F} , differentiable with continuous derivatives. Assume \mathcal{U} is a finite witness set for set \mathcal{F} , and let the sequence x_t be defined by the Stochastic WDS algorithm above. Further assume there is $\pi > 0$ such that $P_t(u) > \pi$ for all $u \in \mathcal{U} \cap \mathcal{D}_{x_{t-1}}$. All accumulation points of the sequence x_t are then solutions of problem (14) with probability 1.*

Proof We want to evaluate the probability of event Q comprising all sequences of selected directions (u_1, u_2, \dots) leading to a situation where x_t has an accumulation point x^* that is not a solution of problem (14).

For each sequence of directions (u_1, u_2, \dots) , the sequence $f(x_t)$ is increasing and bounded. It converges to $f^* = \sup_t f(x_t)$. We have $f(x^*) = f^*$ because f is continuous. By Theorem 6, there is a direction $u \in \mathcal{U}$ such that $f^*(x^*, u) > f^*$ and $\phi(x^*, u) > 0$. Let x_{k_t} be a subsequence converging to x^* . Thanks to the continuity of ϕ , f^* and ∇f , there is a t_0 such that $f^*(x_{k_t}, u) > f^*$ and $\phi(x_{k_t}, u) > 0$ for all $k_t > t_0$.

Choose $\varepsilon > 0$ and let $Q_T \subset Q$ contain only sequences of directions such that $t_0 = T$. For any $k_t > T$, we know that $\phi(x_{k_t}, u) > 0$ which means $u \in \mathcal{U} \cap \mathcal{D}_{x_{k_t}}$. We also know that $u_{k_t} \neq u$ because we would otherwise obtain a contradiction $f(x_{k_t+1}) = f^*(x_{k_t}, u) > f^*$. The probability of selecting such a u_{k_t} is therefore smaller than $(1 - \pi)$. The probability that this happens simultaneously for N distinct $k_t \geq T$ is smaller than $(1 - \pi)^N$ for any N . We get $P(Q_T) \leq \varepsilon/T^2$ by choosing N large enough.

Then we have $P(Q) = \sum_T P(Q_T) \leq \varepsilon (\sum_T 1/T^2) = K\varepsilon$. Hence $P(Q) = 0$ because we can choose ε as small as we want, We can therefore assert with probability 1 that all accumulation points of sequence x_t are solutions. ■

This condition on the distributions P_t is unfortunately too restrictive. The PROCESS and RE-PROCESS iterations of the Online LASVM algorithm (Section 3.2) only exploit directions from very specific subsets.

On the other hand, the Online LASVM algorithm only ensures that any remaining feasible direction at time T will eventually be selected with probability 1. Yet it is challenging to mathematically express that there is no coupling between the subset of time points t corresponding to a subsequence converging to a particular accumulation point, and the subset of time points t corresponding to the iterations where specific feasible directions are selected.

This problem also occurs in the deterministic Generalized SMO algorithm (Section 3.1). An asymptotic convergence proof (Lin, 2001) only exist for the important case of the SVM QP problem using a specific direction selection strategy. Following Keerthi and Gilbert (2002), we bypass this technical difficulty by defining a notion of approximate optimum and proving convergence in finite time. It is then easy to discuss the properties of the limit point.

A.5 Approximate Witness Direction Search

Definition 12 *Given a finite witness family \mathcal{U} and the tolerances $\kappa > 0$ and $\tau > 0$, we say that x is a $\kappa\tau$ -approximate solution of problem (14) when the following condition is verified:*

$$\forall u \in \mathcal{U}, \quad \phi(x, u) \leq \kappa \text{ or } u' \nabla f(x) \leq \tau.$$

A vector $u \in \mathbb{R}_n$ such that $\phi(x, u) > \kappa$ and $u' \nabla f(x) > \tau$ is called a $\kappa\tau$ -violating direction in point x .

This definition is inspired by assertion (iii) in Theorem 6. The definition demands a *finite* witness family because this leads to proposition 13 establishing that $\kappa\tau$ -approximate solutions indicate the location of actual solutions when κ and τ tend to zero.

Proposition 13 *Let \mathcal{U} be a finite witness family for bounded convex set \mathcal{F} . Consider a sequence $x_t \in \mathcal{F}$ of $\kappa_t\tau_t$ -approximate solutions of problem (14) with $\tau_t \rightarrow 0$ and $\kappa_t \rightarrow 0$. The accumulation points of this sequence are solutions of problem (14).*

Proof Consider an accumulation point x^* and a subsequence x_{k_t} converging to x^* . Define function

$$(x, \tau, \kappa) \mapsto \psi(x, \tau, \kappa, u) = (u' \nabla f(x) - \tau) \max \{0, \phi(x, u) - \kappa\}$$

such that u is a $\kappa\tau$ -violating direction if and only if $\psi(x, \kappa, \tau, u) > 0$. Function ψ is continuous thanks to Theorem 9, proposition 10 and to the continuity of ∇f . Therefore, we have $\psi(x_{k_t}, \kappa_{k_t}, \tau_{k_t}, u) \leq 0$ for all $u \in \mathcal{U}$. Taking the limit when $k_t \rightarrow \infty$ gives $\psi(x^*, 0, 0, u) \leq 0$ for all $u \in \mathcal{U}$. Theorem 6 then states that x^* is a solution. ■

The following algorithm introduces the two tolerance parameters $\tau > 0$ and $\kappa > 0$ into the Stochastic Witness Direction Search algorithm.

Approximate Stochastic Witness Direction Search

- 1) Find an initial feasible point $x_0 \in \mathcal{F}$.
- 2) For each $t = 1, 2, \dots$,
 - 2a) Draw a direction $u_t \in \mathcal{U}$ from a probability distribution P_t
 - 2b) If u_t is a $\kappa\tau$ -violating direction,

$$x_t \leftarrow \operatorname{argmax} f(x) \text{ under } x \in \{x_{t-1} + \lambda u_t \in \mathcal{F}, \lambda \geq 0\}$$
 otherwise

$$x_t \leftarrow x_{t-1}.$$

The successive search directions u_t are drawn from some unspecified distributions P_t defined on \mathcal{U} . Proposition 16 establishes that this algorithm always converges to some $x^* \in \mathcal{F}$ after a finite number of steps, regardless of the selected directions (u_t). The proof relies on the two intermediate results that generalize a lemma proposed by Keerthi and Gilbert (2002) in the case of quadratic functions.

Proposition 14 *If u_t is a $\kappa\tau$ -violating direction in x_{t-1} ,*

$$\phi(x_t, u_t) u_t' \nabla f(x_t) = 0.$$

Proof Let the maximum $f(x_t) = f^*(x_{t-1}, u_t)$ be attained in $x_t = x_{t-1} + \lambda^* u_t$ with $0 \leq \lambda^* \leq \phi(x_{t-1}, u_t)$. We know that $\lambda^* \neq 0$ because u_t is $\kappa\tau$ -violating and proposition 2 implies $f^*(x_{t-1}, u_t) > f(x_{t-1})$. If λ^* reaches its upper bound, $\phi(x_t, u_t) = 0$. Otherwise x_t is an unconstrained maximum and $u_t' \nabla f(x_t) = 0$. ■

Proposition 15 *There is a constant $K > 0$ such that*

$$\forall t, f(x_t) - f(x_{t-1}) \geq K \|x_t - x_{t-1}\|.$$

Proof The relation is obvious when u_t is not a $\kappa\tau$ -violating direction in x_{t-1} . Otherwise let the maximum $f(x_t) = f^*(x_{t-1}, u_t)$ be attained in $x_t = x_{t-1} + \lambda^* u_t$. Let $\lambda = \nu\lambda^*$ with $0 < \nu \leq 1$. Since x_t is a maximum,

$$f(x_t) - f(x_{t-1}) = f(x_{t-1} + \lambda^* u_t) - f(x_{t-1}) \geq f(x_{t-1} + \lambda u_t) - f(x_{t-1}).$$

Let H be the maximum over \mathcal{F} of the norm of the Hessian of f . A Taylor expansion with the Cauchy remainder gives

$$\left| f(x_{t-1} + \lambda u_t) - f(x_{t-1}) - \lambda u_t' \nabla f(x_{t-1}) \right| \leq \frac{1}{2} \lambda^2 \|u_t\|^2 H$$

or, more specifically,

$$f(x_{t-1} + \lambda u_t) - f(x_{t-1}) - \lambda u_t' \nabla f(x_{t-1}) \geq -\frac{1}{2} \lambda^2 \|u_t\|^2 H.$$

Combining these inequalities yields

$$f(x_t) - f(x_{t-1}) \geq f(x_{t-1} + \lambda u_t) - f(x_{t-1}) \geq \lambda u_t' \nabla f(x_{t-1}) - \frac{1}{2} \lambda^2 \|u_t\|^2 H.$$

Recalling $u_t' \nabla f(x_{t-1}) > \tau$, and $\lambda \|u_t\| = \nu \|x_t - x_{t-1}\|$, we obtain

$$f(x_t) - f(x_{t-1}) \geq \|x_t - x_{t-1}\| \left(\nu \frac{\tau}{U} - \nu^2 \frac{1}{2} D H \right)$$

where $U = \max_{u \in \mathcal{U}} \|u\|$ and D is the diameter of the compact convex \mathcal{F} .

Choosing $\nu = \min\left(1, \frac{\tau}{UDH}\right)$ then gives the desired result. ■

Proposition 16 *Assume \mathcal{U} is a finite witness set for set \mathcal{F} . The Approximate Stochastic WDS algorithm converges to some $x^* \in \mathcal{F}$ after a finite number of steps.*

Proof Sequence $f(x_t)$ converges because it is increasing and bounded. Therefore it satisfies Cauchy's convergence criterion:

$$\forall \varepsilon > 0, \exists t_0, \forall t_2 > t_1 > t_0, \\ f(x_{t_2}) - f(x_{t_1}) = \sum_{t_1 < t \leq t_2} f(x_t) - f(x_{t-1}) < \varepsilon.$$

Using proposition 15, we can write

$$\forall \varepsilon > 0, \exists t_0, \forall t_2 > t_1 > t_0, \\ \|x_{t_2} - x_{t_1}\| \leq \sum_{t_1 < t \leq t_2} \|x_t - x_{t-1}\| \leq \sum_{t_1 < t \leq t_2} \frac{f(x_t) - f(x_{t-1})}{K} < \frac{\varepsilon}{K}.$$

Therefore sequence x_t satisfies Cauchy's condition and converges to some $x^* \in \mathcal{F}$.

Assume this convergence does not occur in a finite time. Since \mathcal{U} is finite, the algorithm exploits at least one direction $u \in \mathcal{U}$ an infinite number of times. Therefore there is a strictly increasing sequence of positive indices k_t such that $u_{k_t} = u$ is $\kappa\tau$ -violating in point x_{k_t-1} . We have then

$\phi(x_{k_t-1}, u) > \kappa$ and $u' \nabla f(x_{k_t-1}) > \tau$. By continuity we have $\phi(x^*, u) \geq \kappa$ and $u' \nabla f(x^*) \geq \tau$. On the other hand, proposition 14 states that $\phi(x_{k_t}, u) u' \nabla f(x_{k_t}) = 0$. By continuity when $t \rightarrow 0$, we obtain the contradiction $\phi(x^*, u) u' \nabla f(x^*) = 0$. ■

In general, proposition 16 only holds for $\kappa > 0$ and $\tau > 0$. Keerthi and Gilbert (2002) assert a similar property for $\kappa = 0$ and $\tau > 0$ in the case of SVMs only. Despite a mild flaw in the final argument of the initial proof, this assertion is correct (Takahashi and Nishi, 2003).

Proposition 16 does not prove that the limit x^* is related to the solution of the optimization problem (14). Additional assumptions on the direction selection step are required. Theorem 17 addresses the deterministic case by considering trivial distributions P_t that always select a $\kappa\tau$ -violating direction if such directions exist. Theorem 18 addresses the stochastic case under mild conditions on the distribution P_t .

Theorem 17 *Let the concave function f defined on the compact convex set \mathcal{F} be twice differentiable with continuous second derivatives. Assume \mathcal{U} is a finite witness set for set \mathcal{F} , and let the sequence x_t be defined by the Approximate Stochastic WDS algorithm above. Assume that step (2a) always selects a $\kappa\tau$ -violating direction in x_{t-1} if such directions exist. Then x_t converges to a $\kappa\tau$ -approximate solution of problem (14) after a finite number of steps.*

Proof Proposition 16 establishes that there is t_0 such that $x_t = x^*$ for all $t \geq t_0$. Assume there is a $\kappa\tau$ -violating direction in x^* . For any $t > t_0$, step (2a) always selects such a direction, and step (2b) makes x_t different from $x_{t-1} = x^*$. This contradicts the definition of t_0 . Therefore there are no $\kappa\tau$ -violating direction in x^* and x^* is a $\kappa\tau$ -approximate solution. ■

Example (SMO) The SMO algorithm (Section 3.1) is⁴ an Approximate Stochastic WDS that always selects a $\kappa\tau$ -violating direction when one exists. Therefore Theorem 17 applies.

Theorem 18 *Let the concave function f defined on the compact convex set \mathcal{F} be twice differentiable with continuous second derivatives. Assume \mathcal{U} is a finite witness set for set \mathcal{F} , and let the sequence x_t be defined by the Approximate Stochastic WDS algorithm above. Let p_t be the conditional probability that u_t is $\kappa\tau$ -violating in x_{t-1} given that \mathcal{U} contains such directions. Assume that $\limsup p_t > 0$. Then x_t converges with probability one to a $\kappa\tau$ -approximate solution of problem (14) after a finite number of steps.*

Proof Proposition 16 establishes that for each sequence of selected directions u_t , there is a time t_0 and a point $x^* \in \mathcal{F}$ such that $x_t = x^*$ for all $t \geq t_0$. Both t_0 and x^* depend on the sequence of directions (u_1, u_2, \dots) .

We want to evaluate the probability of event Q comprising all sequences of directions (u_1, u_2, \dots) leading to a situation where there are $\kappa\tau$ -violating directions in point x^* . Choose $\varepsilon > 0$ and let $Q_T \subset Q$ contain only sequences of decisions (u_1, u_2, \dots) such that $t_0 = T$.

Since $\limsup p_t > 0$, there is a subsequence k_t such that $p_{k_t} \geq \pi > 0$. For any $k_t > T$, we know that \mathcal{U} contains $\kappa\tau$ -violating directions in $x_{k_t-1} = x^*$. Direction u_{k_t} is not one of them because this

4. Strictly speaking we should introduce the tolerance $\kappa > 0$ into the SMO algorithm. We can also claim that (Keerthi and Gilbert, 2002; Takahashi and Nishi, 2003) have established proposition 16 with $\kappa = 0$ and $\tau > 0$ for the specific case of SVMs. Therefore Theorems 17 and 18 remain valid.

would make x_{k_t} different from $x_{k_t-1} = x^*$. This occurs with probability $1 - p_{k_t} \leq 1 - \pi < 1$. The probability that this happens simultaneously for N distinct $k_t > T$ is smaller than $(1 - \pi)^N$ for any N . We get $P(Q_T) \leq \epsilon/T^2$ by choosing N large enough.

Then we have $P(Q) = \sum_T P(Q_T) \leq \epsilon (\sum_T 1/T^2) = K\epsilon$. Hence $P(Q) = 0$ because we can choose ϵ as small as we want. We can therefore assert with probability 1 that \mathcal{U} contains no $\kappa\tau$ -violating directions in point x^* . ■

Example (LASVM) The LASVM algorithm (Section 3.2) is⁵ an Approximate Stochastic WDS that alternates two strategies for selecting search directions: PROCESS and REPROCESS. Theorem 18 applies because $\limsup p_t > 0$.

Proof Consider an arbitrary iteration T corresponding to a REPROCESS.

Let us define the following assertions:

- A – There are τ -violating pairs (i, j) with both $i \in S$ and $j \in S$.
- B – A is false, but there are τ -violating pairs (i, j) with either $i \in S$ or $j \in S$.
- C – A and B are false, but there are τ -violating pairs (i, j) .
- Q_t – Direction u_t is τ -violating in x_{t-1} .

A reasoning similar to the convergence discussion in Section 3.2 gives the following lower bounds (where n is the total number of examples).

$$\begin{aligned} P(Q_T|A) &= 1 \\ P(Q_T|B) &= 0 \quad P(Q_{T+1}|B) \geq n^{-1} \\ P(Q_T|C) &= 0 \quad P(Q_{T+1}|C) = 0 \quad P(Q_{T+2}|C) = 0 \quad P(Q_{T+3}|C) \geq n^{-2}. \end{aligned}$$

Therefore

$$\begin{aligned} P(Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+3} | A) &\geq n^{-2} \\ P(Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+3} | B) &\geq n^{-2} \\ P(Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+3} | C) &\geq n^{-2}. \end{aligned}$$

Since $p_t = P(Q_t | A \cup B \cup C)$ and since the events A , B , and C are disjoint, we have

$$p_T + p_{T+1} + p_{T+2} + p_{T+3} \geq P(Q_T \cup Q_{T+1} \cup Q_{T+2} \cup Q_{T+3} | A \cup B \cup C) \geq n^{-2}.$$

Therefore $\limsup p_t \geq \frac{1}{4} n^{-2}$. ■

Example (LASVM + Gradient Selection) The LASVM algorithm with Gradient Example Selection remains an Approximate WDS algorithm. Whenever Random Example Selection has a non zero probability to pick a τ -violating pair, Gradient Example Selection picks the a τ -violating pair with maximal gradient with probability one. Reasoning as above yields $\limsup p_t \geq 1$. Therefore Theorem 18 applies and the algorithm converges to a solution of the SVM QP problem.

Example (LASVM + Active Selection + Randomized Search) The LASVM algorithm with Active Example Selection remains an Approximate WDS algorithm. However it does not necessarily verify the conditions of Theorem 18. There might indeed be τ -violating pairs that do not involve the example closest to the decision boundary.

However, convergence occurs when one uses the Randomized Search method to select an example near the decision boundary. There is indeed a probability greater than $1/n^M$ to draw a sample

5. See footnote 4 discussing the tolerance κ in the case of SVMs.

containing M copies of the same example. Reasoning as above yields $\limsup p_t \geq \frac{1}{4} n^{-2M}$. Therefore, Theorem 18 applies and the algorithm eventually converges to a solution of the SVM QP problem.

In practice this convergence occurs very slowly because it involves very rare events. On the other hand, there are good reasons to prefer the intermediate kernel classifiers visited by this algorithm (see Section 4).

References

- M. A. Aizerman, É. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
- N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.
- G. Bakır, L. Bottou, and J. Weston. Breaking SVM complexity with cross-training. In Lawrence Saul, Bernhard Schölkopf, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 81–88. MIT Press, 2005.
- A. Bordes and L. Bottou. The Huller: a simple and efficient online SVM. In *Proceedings of the 16th European Conference on Machine Learning (ECML2005)*, Lecture Notes in Artificial Intelligence, to appear. Springer, 2005.
- L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Conference B: Computer Vision & Image Processing.*, volume 2, pages 77–82, Jerusalem, October 1994. IEEE.
- L. Bottou and Y. LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005.
- C. Campbell, N. Cristianini, and A. J. Smola. Query learning with large margin classifiers. In *Proceedings of ICML'2000*, 2000.
- G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Processing Systems*, 2001.
- N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 241–248. MIT Press, Cambridge, MA, 2005.
- C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. Technical report, Computer Science and Information Engineering, National Taiwan University, 2001-2004. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- D. Cohn, L. Atlas, and R. Ladner. Training connectionist networks with queries and selective sampling. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufmann.

- R. Collobert and S. Bengio. SVM Torch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.
- R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of SVMs for very large scale problems. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
- C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
- K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
- K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.
- N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK, 2000.
- C. Domingo and O. Watanabe. MadaBoost: a modification of AdaBoost. In *Proceedings of the 13th Annual Conference on Computational Learning Theory, COLT'00*, pages 180–189, 2000.
- B. Eisenberg and R. Rivest. On the sample complexity of PAC learning using random and chosen examples. In M. Fulk and J. Case, editors, *Proceedings of the Third Annual ACM Workshop on Computational Learning Theory*, pages 154–162, San Mateo, CA, 1990. Kaufmann.
- V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.
- Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In J. Shavlik, editor, *Machine Learning: Proceedings of the Fifteenth International Conference*, San Francisco, CA, 1998. Morgan Kaufmann.
- T.-T. Frieß, N. Cristianini, and C. Campbell. The kernel Adatron algorithm: a fast and simple learning procedure for support vector machines. In J. Shavlik, editor, *15th International Conf. Machine Learning*, pages 188–196. Morgan Kaufmann Publishers, 1998. See (Cristianini and Shawe-Taylor, 2000, section 7.2) for an updated presentation.
- C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.
- H.-P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik. Parallel support vector machines: The Cascade SVM. In Lawrence Saul, Bernhard Schölkopf, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.
- I. Guyon, B. Boser, and V. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In S. J. Hanson, J. D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 147–155. Morgan Kaufmann, San Mateo, CA, 1993.
- T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

- S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46:351–360, 2002.
- Y. Li and P. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46:361–387, 2002.
- C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, 2001.
- N. Littlestone and M. Warmuth. Relating data compression and learnability. Technical report, University of California Santa Cruz, 1986.
- G. Loosli, S. Canu, S.V.N. Vishwanathan, A. J. Smola, and M. Chattopadhyay. Une boîte à outils rapide et simple pour les SVM. In Michel Liquière and Marc Sebban, editors, *CAp 2004 - Confrence d'Apprentissage*, pages 113–128. Presses Universitaires de Grenoble, 2004. ISBN 9-782706-112249.
- D. J. C. MacKay. Information based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1992.
- N. Murata and S.-I. Amari. Statistical analysis of learning dynamics. *Signal Processing*, 74(1): 3–28, 1999.
- N. J. Nilsson. *Learning machines: Foundations of Trainable Pattern Classifying Systems*. McGraw-Hill, 1965.
- A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.
- J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.
- F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
- G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 839–846. Morgan Kaufmann, June 2000.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, 1986.
- I. Steinwart. Sparseness of support vector machines—some asymptotically sharp bounds. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

- N. Takahashi and T. Nishi. On termination of the SMO algorithm for support vector machines. In *Proceedings of International Symposium on Information Science and Electrical Engineering 2003 (ISEE 2003)*, pages 187–190, November 2003.
- S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, San Francisco, California, 2000. Morgan Kaufmann.
- I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Very large SVM training using core vector machines. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT'05)*. Society for Artificial Intelligence and Statistics, 2005.
- V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin, 1982.
- V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.
- V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.
- V. N. Vapnik, T. G. Glaskova, V. A. Koscheev, A. I. Mikhailski, and A. Y. Chervonenkis. *Algorithms and Programs for Dependency Estimation*. Nauka, 1984. In Russian.
- S. V. N. Vishwanathan, A. J. Smola, and M. Narasimha Murty. SimpleSVM. In *Proceedings of ICML 2003*, pages 760–767, 2003.
- J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados*, pages 413–420. Society for Artificial Intelligence and Statistics, 2005.
- G. Zoutendijk. *Methods of Feasible Directions*. Elsevier, 1960.

Managing Diversity in Regression Ensembles

Gavin Brown

Jeremy L. Wyatt

Peter Tiño

School of Computer Science

University of Birmingham

Birmingham, UK, B15 2TT

G.BROWN@CS.BHAM.AC.UK

J.L.WYATT@CS.BHAM.AC.UK

P.TINO@CS.BHAM.AC.UK

Editor: Yoshua Bengio

Abstract

Ensembles are a widely used and effective technique in machine learning—their success is commonly attributed to the degree of disagreement, or ‘diversity’, within the ensemble. For ensembles where the individual estimators output crisp class labels, this ‘diversity’ is not well understood and remains an open research issue. For ensembles of regression estimators, the diversity can be exactly formulated in terms of the covariance between individual estimator outputs, and the optimum level is expressed in terms of a *bias-variance-covariance* trade-off. Despite this, most approaches to learning ensembles use heuristics to encourage the right degree of diversity. In this work we show how to explicitly control diversity through the error function. The first contribution of this paper is to show that *by taking the combination mechanism for the ensemble into account we can derive an error function for each individual that balances ensemble diversity with individual accuracy*. We show the relationship between this error function and an existing algorithm called *negative correlation learning*, which uses a heuristic penalty term added to the mean squared error function. It is demonstrated that these methods control the bias-variance-covariance trade-off systematically, and can be utilised with any estimator capable of minimising a quadratic error function, for example MLPs, or RBF networks. As a second contribution, we derive a strict upper bound on the coefficient of the penalty term, which holds for any estimator that can be cast in a generalised linear regression framework, with mild assumptions on the basis functions. Finally we present the results of an empirical study, showing significant improvements over simple ensemble learning, and finding that this technique is competitive with a variety of methods, including boosting, bagging, mixtures of experts, and Gaussian processes, on a number of tasks.

Keywords: ensemble, diversity, regression estimators, neural networks, hessian matrix, negative correlation learning

1. Introduction

The last decade has seen a frenzy of work in so-called *ensemble learning systems*. These are groups of machine learning systems where each learner provides an estimate of a target variable; these estimates are combined in some fashion, hopefully reducing the generalisation error compared to a single learner. The target can be categorical (classification ensembles) or continuous (regression ensembles). The multiple estimates are integrated via a combination function, commonly *majority voting* for classification and a *linear combination* for regression. It is well appreciated in both cases that the individual estimators should exhibit different patterns of generalisation—the very simple intuitive explanation is that a million identical estimators are obviously no better than a single

estimator of the same form. Much research has gone into how to encourage this error “diversity”—most commonly manipulating the training data, providing each learner with a different subset of patterns or features (see Brown et al. (2005a) for a recent survey). The main point to note here is that when our estimators output crisp class labels, there is no agreed definition of diversity, and it remains an open research question (Kuncheva and Whitaker (2003)). The problem is somewhat easier if we have estimators that give posterior probabilities, in which case the effect of estimator correlations on classification error rate has been investigated by Tumer and Ghosh (1995) and Fumera and Roli (2003), though there remain several open questions on this topic.

A commonly overlooked point for regression ensembles is that this “diversity” can be explicitly quantified and measured. The *bias-variance-covariance* decomposition from Ueda and Nakano (1996) breaks the mean squared error (MSE) into three components. Quite simply, whereas in a single regression estimator we have the well known bias-variance *two-way* trade-off, in an ensemble of regressors we have the bias-variance-covariance *three-way* trade-off. The optimum “diversity” is that which optimally balances the components to reduce the overall MSE. In this article we focus on *negative correlation (NC) learning*, a successful neural network ensemble learning technique developed in the evolutionary computation literature (Liu (1998)). In a statistical framework, we show that NC uses a penalty coefficient to *explicitly* alter the emphasis on the variance and covariance portions of the MSE. Setting a zero coefficient corresponds to independently training the estimators; a higher coefficient introduces more emphasis on covariance, and at a particular value it corresponds to treating the entire ensemble as a single learning unit. This is an explicit *management* of the ensemble diversity. We will describe how the ensemble error gradient can be broken into a number of individually understandable components, and that NC exploits this to blend smoothly between a group of independent learners and a single large learner, finding the optimal point *between the two*. We will prove an upper bound on the penalty coefficient, provide guidance on how to set it optimally, and show empirical support that this guidance is useful. The NC framework is *applicable to any nonlinear regression estimator* minimising the MSE; we show examples using multi-layer perceptrons and radial basis function networks as the base estimators.

The structure of this article is as follows. We begin in Section 2 with a summary of the underlying theory of regression ensemble learning, describing why there exists a trade-off between ensemble diversity and individual estimator accuracy. We then consider in Section 3 how we might derive an error function that is capable of optimising this trade-off *explicitly*. We do this and note that it can be shown as equivalent to an existing heuristic technique, *negative correlation (NC) learning*. We continue in Section 4 with an introduction to NC learning, summarising the assumptions and properties as published in the original work. Here we provide a statistical interpretation of NC, and derive a strict upper bound on its penalty coefficient—we empirically validate this bound in Sections 5 and 6. Finally in Section 7 we summarise the implications of this work in a broad context.

2. Ensemble Learning for Regression

In this section we review the bias-variance decomposition (Geman et al. (1992)), using it as a vehicle to introduce our notation; we then show how this decomposition naturally extends to a *bias-variance-covariance* decomposition (Ueda and Nakano (1996)) when using an ensemble of regression estimators.

2.1 The Bias-Variance Decomposition

We have a data set of input vectors and output scalars, $z = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$, with each element drawn from a random variable Z defined over an unknown distribution $p(\mathbf{x}, t)$. It should be noted that for brevity, and without loss of generality, we have assumed a noise level of zero in the data.¹ The learning problem is to use the set z to approximate the correct mapping from input to output. For this purpose we use a parameterized estimator f , whose set of parameters \mathbf{w} determine how well it approximates the mapping. We would like to find the set of parameters \mathbf{w} that minimise the expected mean squared error,

$$e(f) = \int (f(\mathbf{x}; \mathbf{w}) - t)^2 p(\mathbf{x}, t) d(\mathbf{x}, t). \quad (1)$$

Unfortunately we do not have access to the true distribution $p(\mathbf{x}, t)$, so we approximate this integral with a summation over the data set z ,

$$e(f) \approx \frac{1}{N} \sum_{n=1}^N (f(\mathbf{x}_n; \mathbf{w}) - t_n)^2, \quad (\mathbf{x}_n, t_n) \in z. \quad (2)$$

We do not necessarily want a set of parameters \mathbf{w} that give us zero error on z ; this is because z is only a *sample* from the true distribution, and if we tune \mathbf{w} precisely to z then the estimator f may not perform well on future data (we overfitted). However, if we do not tune \mathbf{w} just *enough*, then we may again not perform well in the future (we underfitted). This is explicitly formulated in the *bias-variance* decomposition (Geman et al. (1992)). Note that from this point forward, in place of the integral notation in Equation (1), we use the shorthand expectation operator $E\{\cdot\}$; additionally we will omit the input and parameter vectors, so where it is unambiguous, instead of $f(\mathbf{x}; \mathbf{w})$, we write simply f . The bias-variance decomposition is

$$\begin{aligned} E\{(f - t)^2\} &= (E\{f\} - t)^2 + E\{(f - E\{f\})^2\} \\ &= \text{bias}(f)^2 + \text{variance}(f). \end{aligned} \quad (3)$$

The decomposition is a property of the *generalisation* error; these two components have to be balanced against each other for best performance. Now let us imagine that instead of a single estimator f , we have a collection of them: f_1, \dots, f_M , each f_i has its own parameter vector \mathbf{w}_i , and M is the total number of estimators. We then train each individual f_i separately, using Equation (2) as the error function; once this is accomplished, the outputs of the individuals are *combined* to give the *ensemble output* for any new datapoint \mathbf{x} . The simplest possible combination mechanism is to take a uniformly weighted average, so the output of the ensemble is

$$\bar{f}(\mathbf{x}; \mathbf{w}_1, \dots, \mathbf{w}_M) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}; \mathbf{w}_i). \quad (4)$$

The ensemble \bar{f} can obviously be seen as an estimator in its own right; it will therefore have a bias-variance decomposition; However it transpires that, for this class of estimator, it can be extended to a bias-variance-*covariance* decomposition.

1. In the case of a non-zero noise component, t in the decomposition would be replaced by its expected value $E_T\{t\}$, and a constant (irreducible) term σ^2 would be added, representing the variance of the noise.

2.2 The Bias-Variance-Covariance Decomposition

Treating the ensemble as a single learning unit, its bias-variance decomposition can be formulated as

$$\begin{aligned} E\{(\bar{f} - t)^2\} &= (E\{\bar{f}\} - t)^2 + E\{(\bar{f} - E\{\bar{f}\})^2\} \\ &= \text{bias}(\bar{f})^2 + \text{variance}(\bar{f}). \end{aligned} \tag{5}$$

We will now consider how the bias-variance decomposition for an ensemble can be extended (Ueda and Nakano (1996)).² From this point forward, it should be noted that the expectation operator is subtly different to that in the decomposition for a single estimator. We redefine our random variable Z as a set $Z = (Z_1, \dots, Z_M)$, so the i th estimator is trained with a training set z_i drawn from its own random variable Z_i . It should be noted that Z_i potentially may be identical for all i , or not. If the training data is identical for two machines i and j , it does not imply that the expected values $E\{f_i\}$ and $E\{f_j\}$ are equal, since other differences may be present between machines i and j , i.e. in the training procedures, or the models. Finally, we note that although the decomposition presented below does hold for non-uniformly weighted ensembles, we restrict our analysis to the uniform case, as it corresponds to the simple average combination technique used commonly in practice. To aid our exposition now, we define three concepts. The first concept is $\overline{\text{bias}}$, the averaged bias of the ensemble members,

$$\overline{\text{bias}} = \frac{1}{M} \sum_i (E\{f_i\} - t). \tag{6}$$

The second is $\overline{\text{var}}$, the averaged variance of the ensemble members,

$$\overline{\text{var}} = \frac{1}{M} \sum_i E\{(f_i - E\{f_i\})^2\}. \tag{7}$$

The third is $\overline{\text{covar}}$, the averaged covariance of the ensemble members,

$$\overline{\text{covar}} = \frac{1}{M(M-1)} \sum_i \sum_{j \neq i} E\{(f_i - E\{f_i\})(f_j - E\{f_j\})\}. \tag{8}$$

We then have

$$E\{(\bar{f} - t)^2\} = \overline{\text{bias}}^2 + \frac{1}{M} \overline{\text{var}} + \left(1 - \frac{1}{M}\right) \overline{\text{covar}}. \tag{9}$$

What does this decomposition tell us? It illustrates that in addition to the bias and variance of the individual estimators, the generalisation error of an ensemble also depends on the *covariance* between the individuals. This raises the interesting issue of why we should ever train ensemble members separately; why shouldn't we try to find some way to capture the effect of the covariance in the error function? Given the decomposition (9), it is not immediately obvious what form this should take—this will be our next topic for consideration.

2. It is interesting to note that this was the first appearance of the decomposition only for the ML literature—in fact an equivalent decomposition can be found in Markowitz (1952), which was instrumental for modern financial portfolio theory, and subsequently won the 1990 Nobel Prize for Economics.

3. How Can We Optimise Diversity with an Error Function?

For a single regression estimator, generalisation error is determined by a two-way *bias-variance* trade-off; for an ensemble of regression estimators, the ‘diversity’ issue is simply a *three-way bias-variance-covariance* trade-off. We know how to quantify diversity, but we have not yet considered how to achieve it and balance it against individual accuracy—the fundamental issue of ensemble learning. The decompositions we have considered so far consist of integrals over *all possible data sets* of a fixed size—we require a computable approximation to these in order to minimise an error function on a limited data set. It turns out that another decomposition in the literature, significantly more well-known, provides the missing link. We will review this decomposition and its relation to the ones we have already considered, then show how we can use it to train an ensemble whilst controlling the bias-variance-covariance trade-off.

3.1 The Ambiguity Decomposition

Krogh and Vedelsby (1995) showed that *at a single arbitrary datapoint, the quadratic error of the ensemble estimator is guaranteed to be less than or equal to the weighted average quadratic error of the component estimators,*

$$(f_{ens} - t)^2 = \sum_i c_i (f_i - t)^2 - \sum_i c_i (f_i - f_{ens})^2. \quad (10)$$

where t is the target value of an arbitrary datapoint, $\sum_i c_i = 1$, $c_i \geq 0$, and f_{ens} is the convex combination of the M component estimators $f_{ens} = \sum_{i=1}^M c_i f_i$. Preceding the bias-variance-covariance decomposition, this was a very encouraging result for ensemble research, providing a very simple expression for the effect of error correlations in an ensemble. The decomposition is made up of two terms. The first, $\sum_i c_i (f_i - t)^2$, is the weighted average error of the individuals. The second, $\sum_i c_i (f_i - f_{ens})^2$ is referred to as the *Ambiguity*, measuring the amount of variability among the ensemble member answers for this particular (\mathbf{x}, t) pair. The trade-off between these two determines how well the ensemble performs at this datapoint.

We have now seen two decompositions, Equation (9) and Equation (10), expressing the effect of correlations on ensemble error in two different ways. It is therefore sensible to ask what the relationship is between these two. The very similar structure of the two decompositions (5) and (10) is no coincidence; the proofs are virtually identical (Brown et al. (2005a)); see also Hansen (2000) for an alternative treatment of this relationship. Assuming a uniform weighting, we substitute the right hand side of equation (10) into the left hand side of equation (9), giving us

$$E\left\{\frac{1}{M} \sum_i (f_i - t)^2 - \frac{1}{M} \sum_i (f_i - \bar{f})^2\right\} = \overline{bias}^2 + \frac{1}{M} \overline{var} + \left(1 - \frac{1}{M}\right) \overline{covar}. \quad (11)$$

What portions of the bias-variance-covariance decomposition correspond to the Ambiguity term? After some manipulations (see Appendix B for details) we can show

$$E\left\{\frac{1}{M} \sum_i (f_i - t)^2\right\} = \overline{bias}^2 + \Omega \quad (12)$$

$$E\left\{\frac{1}{M} \sum_i (f_i - \bar{f})^2\right\} = \Omega - \left[\frac{1}{M} \overline{var} + \left(1 - \frac{1}{M}\right) \overline{covar}\right]. \quad (13)$$

where Ω is the interaction between the two sides,

$$\Omega = \overline{\text{var}} + \frac{1}{M} \sum_i (E\{f_i\} - E\{\bar{f}\})^2. \tag{14}$$

Since the Ω is present in both sides, when we combine them by subtracting the Ambiguity in equation (13), from the average MSE in equation (12), the Ω s cancel out, and we get the original bias-variance-covariance decomposition back, as in the RHS of equation (11). This Ω term is the average variance of the estimators, plus the average squared deviation of the expectations of the individuals from the expectation of the ensemble. The fact that the Ω term exists illustrates again that we cannot simply maximise diversity without affecting the other parts of the error—in effect, this interaction *quantifies* the diversity trade-off for regression ensembles.

3.2 Using the Decompositions to Optimise Diversity

In a simple ensemble, the norm is to train learners separately—the i th member of the ensemble would have the error function³

$$e_i = \frac{1}{2}(f_i - t)^2. \tag{15}$$

In light of the decompositions we have seen, this is rather odd. Why wouldn't we want to directly minimise the *full* ensemble error?

$$e_{ens} = \frac{1}{2}(\bar{f} - t)^2 = \frac{1}{M} \sum_i \frac{1}{2}(f_i - t)^2 - \frac{1}{M} \sum_i \frac{1}{2}(f_i - \bar{f})^2. \tag{16}$$

One easy answer to this is that we are adopting the “division of labor” approach, simplifying the learning problem by breaking it into M smaller problems. However, according to Equation (11), this error function should account for the bias, the variance, and critically also the covariance of the ensemble. The point to remember is that these components should be *balanced* against each another. Given the relationship shown in Equation (12) and Equation (13), we could imagine a “diversity-encouraging” error function of the form

$$e_i^{div} = \frac{1}{M} \sum_i \frac{1}{2}(f_i - t)^2 - \kappa \frac{1}{M} \sum_i \frac{1}{2}(f_i - \bar{f})^2. \tag{17}$$

where κ is a scaling coefficient in $[0, 1]$ and allows us to vary the emphasis on the covariance component. If we adopt a gradient descent procedure for training, we note

$$\frac{\partial e_i^{div}}{\partial f_i} = \frac{1}{M} [(f_i - t) - \kappa(f_i - \bar{f})]. \tag{18}$$

When $\kappa = 0$ here, the gradient of our error function is proportional to the gradient of the error of a single learner, Equation (15). At the other extreme, when $\kappa = 1$, the f_i terms in Equation (18) cancel out, and we have the gradient of the entire ensemble as a single unit,

$$\kappa = 0 \quad , \quad \frac{\partial e_i^{div}}{\partial f_i} = \frac{1}{M} [(f_i - t)] = \frac{1}{M} \frac{\partial e_i}{\partial f_i} \tag{19}$$

$$\kappa = 1 \quad , \quad \frac{\partial e_i^{div}}{\partial f_i} = \frac{1}{M} [(\bar{f} - t)] = \frac{\partial e_{ens}}{\partial f_i}. \tag{20}$$

3. As we will shortly be using a gradient descent procedure, by convention with the existing literature we multiply by $\frac{1}{2}$.

By scaling the κ term we would be able to vary smoothly between the two extremes of training learners separately and training the ensemble as a single unit. Further analysis of these gradient components is provided in Appendix C. Using this scaling parameter corresponds to explicitly varying our emphasis on minimising the covariance term within the ensemble MSE, balancing it against our emphasis on the bias and variance terms; hence we are explicitly *managing* the bias-variance-covariance trade-off. The reader may now justifiably expect an empirical investigation of this error function; however, it conveniently transpires that an existing heuristic method in the literature (derived independently of the observations above) can be shown to be equivalent to this, and has undergone extensive empirical tests showing its utility in a number of domains. The theoretical results we have derived in this section form a solid foundation to explain the success of this technique and link it to others in the literature. We will now consider this, *Negative Correlation Learning*, and show precisely how it relates to the derivations we have provided in this section.

4. Negative Correlation Learning

Negative correlation (NC) learning (Liu (1998)) is a neural network ensemble learning technique developed in the Evolutionary Computation literature. NC has shown a number of empirical successes and varied applications, including regression problems (Yao et al. (2001)), classification problems (McKay and Abbass (2001)), and time-series prediction (Liu (1998)). It has consistently demonstrated significant performance improvements over a simple ensemble system, showing very competitive results with other techniques like mixtures of experts, bagging, and boosting (Liu and Yao (1997); McKay and Abbass (2001)). Though empirical successes have been found with classification problems, it should be noted that the discussion here concerns only the regression case.

4.1 The History

The fact that correlations between ensemble members affects performance has been known for a long time. The first such reference to appear in the machine learning literature was Perrone (1993), showing that we obtain a $\frac{1}{M}$ variance reduction if correlation between learners is zero. The first reference in the literature to explicitly use this idea in a learning algorithm was Rosen (1996), who trained networks sequentially using a penalty and scaling coefficient λ added to the error term,

$$e_i = \frac{1}{2}(f_i - t)^2 + \lambda p_i \quad (21)$$

$$p_i = (f_i - t) \sum_{j=1}^{i-1} (f_j - t). \quad (22)$$

Attempting to extend this work, Liu and Yao (1997) trained the networks in parallel, and used a number of alternative penalty terms⁴ including one where the t is replaced by \bar{f} ,

$$p_i = (f_i - t) \sum_{j \neq i} (f_j - t) \quad (23)$$

$$p_i = (f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}). \quad (24)$$

4. A companion work to this article, Brown et al. (2005b), gives a similar analysis to penalty (23).

The λ parameter is problem-dependent, controlling the trade-off between the objective and penalty terms during the gradient descent training procedure. Figure 1 shows NC using backpropagation to update the network weights. A point to note here is that the authors calculate the gradient using the assumption “that the output of the ensemble \bar{f} has constant value with respect to f_i ” (Liu, 1998, p.29)., i.e.

$$\frac{\partial \bar{f}}{\partial f_i} = 0. \quad (25)$$

Using this, and the penalty Equation (24), the following gradient was derived,

$$e_i = \frac{1}{2}(f_i - t)^2 + \lambda(f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) \quad (26)$$

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) + \lambda \sum_{j \neq i} (f_j - \bar{f}). \quad (27)$$

This is clearly an incorrect assumption—in the next section we will examine the reasoning behind it,

1. Let M be the final number of predictors required.
2. Take a training set $z = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_N, t_N)\}$.
3. For each training pattern in z from $n = 1$ to N do :
 - (a) Calculate $\bar{f} = \frac{1}{M} \sum_i f_i(\mathbf{x}_n)$
 - (b) For each network from $i = 1$ to M do:
 - Perform a *single* update for each weight w in network i , using a learning rate α (set as 0.1 in our experiments), and:

$$\Delta w = -\alpha \left[(f_i(\mathbf{x}_n) - t_n) - \lambda(f_i(\mathbf{x}_n) - \bar{f}) \right] \cdot \frac{\partial f_i}{\partial w}$$
4. Repeat from step 3 for a desired number of iterations.

For any new testing pattern \mathbf{x} , the ensemble output is given by:

$$\bar{f} = \frac{1}{M} \sum_i f_i(\mathbf{x})$$

Figure 1: Pseudocode for negative correlation learning. Note the relationship between the λ term and the γ term in Equation (30).

the implications it brings, and show how NC relates to the error decompositions we have discussed so far.

4.2 A Theoretical Grounding for NC Learning

We would like to provide a rigorous foundation for the NC method, so it seems sensible to observe what happens when we *remove* the assumption of constant \bar{f} . We now introduce a term γ in place of λ , to indicate when we perform the gradient calculations *without the assumption*. Re-deriving the gradient, we have

$$e_i = \frac{1}{2}(f_i - t)^2 + \gamma(f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) \quad (28)$$

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) - \gamma \left[2 \left(1 - \frac{1}{M} \right) (f_i - \bar{f}) \right]. \quad (29)$$

We understand now that λ in fact has the deterministic component $2(1 - \frac{1}{M})$; to avoid confusion we now refer to the parameters in the following context,

$$\lambda = 2\gamma \left(1 - \frac{1}{M} \right). \quad (30)$$

where γ is still a problem-dependent scaling parameter. According to communications with the original authors, the assumption was introduced for two reasons. Firstly because the term $2(1 - \frac{1}{M})$ is a constant for any fixed ensemble of size M , so can be precalculated for efficiency. Secondly, it allowed the appealing property that when $\lambda = 1$, the gradient in Equation (27) reduces

$$\begin{aligned} \frac{\partial e_i}{\partial f_i} &= (f_i - t) + \lambda \sum_{j \neq i} (f_j - \bar{f}) \\ &= (f_i - t) - \lambda(f_i - \bar{f}) \\ &= (\bar{f} - t) \\ &= M \cdot \frac{\partial e_{ens}}{\partial f_i}. \end{aligned} \quad (31)$$

Here it can be seen that the identity $\sum_{j \neq i} (f_j - \bar{f}) = -(f_i - \bar{f})$ was used—the sum of deviations around a mean is equal to zero. However, for this to hold we have to now *violate* the constant \bar{f} assumption, as the sum of deviations around a constant is *not* equal to zero. The reader will see an immediate similarity in (31) to the observations we have made in the previous section, specifically equations (18), (19), and (20). It emerges that by introducing the assumption, and subsequently violating it, the NC gradient becomes proportional to the gradient of the “diversity-encouraging” error function (18) suggested earlier, where we use λ in place of κ ,

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) - \lambda(f_i - \bar{f}) = M \cdot \frac{\partial e_i^{div}}{\partial f_i}. \quad (32)$$

From the observations we have made here, the connection between NC and the Bias-Variance-Covariance decomposition should be apparent. By introducing the assumption (25), NC was inadvertently provided with the missing gradient components that correspond to the variance and covariance terms within the ensemble MSE. It can therefore be concluded that NC succeeds because it trains the individual networks with error functions which more closely approximate the individual’s contribution to ensemble error, than that used by simple ensemble learning. Using the

penalty coefficient, we then balance the trade-off between those individual errors and the ensemble covariance. The relationship to the Ambiguity decomposition is made even more apparent by noting that the penalty term can be rearranged,

$$p_i = (f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) = -(f_i - \bar{f})^2. \quad (33)$$

This leads us to a restatement of the NC error function,

$$e_i = \frac{1}{2}(f_i - t)^2 - \gamma(f_i - \bar{f})^2. \quad (34)$$

Remembering the breakdown of the ensemble error from earlier,

$$e_{ens} = \frac{1}{M} \sum_i \left[\frac{1}{2}(f_i - t)^2 - \frac{1}{2}(f_i - \bar{f})^2 \right], \quad (35)$$

we see that the MSE of an ensemble can be decomposed into a weighted summation, where the i^{th} term is the backpropagation error function plus the NC-learning penalty function, with the γ parameter set at 0.5. Here we note an important point, that there are additional effects that f_i has on Equation (35), that are *not contained* in Equation (34). This is via the \bar{f} term, which obviously depends on f_i , and can be found in each component of the summation in Equation (35). Therefore, simply setting $\gamma = 0.5$ would mean we are not taking account of these effects, and setting $\gamma > 0.5$ allows us to include them and find the appropriate problem-dependent balance for best generalisation. These observations are supported by further gradient analysis in Appendix C.

To summarise, in this section we have shown that there exist two quite different error functions, which yield gradients (18) and (29) differing only in a scalar constant. Each incorporates sufficient information to allow the individual learners to optimise the bias-variance-covariance trade-off. We now understand how NC balances accuracy against diversity; however, we do not yet understand what the *correct* balance is, i.e. how do we set the penalty coefficient? We consider this problem in the next section.

4.3 Understanding and Defining Bounds on the Penalty Coefficient

The original work on NC (Liu and Yao (1997)) showed that a λ value greater than zero can encourage a decrease in covariance, however it is also observed that too high a value can cause a rapid increase in the variance component, causing overall error to be higher. No theoretical explanation was given for this behaviour, and as such we do not yet have a clear picture of the exact dynamics of the parameter. It was stated that the bounds of λ should be $[0, 1]$, based on the following calculation,

$$\begin{aligned} \frac{\partial e_i}{\partial f_i} &= f_i - t + \lambda \sum_{j \neq i} (f_j - \bar{f}) \\ &= f_i - t - \lambda(f_i - \bar{f}) \\ &= f_i - t - \lambda(f_i - \bar{f}) + \lambda t - \lambda t \\ &= (1 - \lambda)(f_i - t) + \lambda(\bar{f} - t). \end{aligned}$$

It is stated: “the value of parameter λ lies inside the range $0 \leq \lambda \leq 1$ so that both $(1 - \lambda)$ and λ have non-negative values” (Liu, 1998, p.29). In practice this bound seemed to be applicable; however,

the justification is questionable, and again here we see the assumption of constant \bar{f} is violated—if constant \bar{f} is assumed, then the deviations around \bar{f} result cannot be used. In this section we provide more concrete theoretical evidence for an upper bound.

The NC penalty term ‘warps’ the error landscape of the network, making the global minimum hopefully easier to locate. However, if the landscape is warped *too* much, it could eliminate any useful gradient information. This state is indicated by the positive-definiteness (PD) of the Hessian matrix. If the Hessian matrix, evaluated at a given point, is non-PD, then the error gradient consists of either a local maximum or a point of inflexion, and we have lost any useful gradient information from our original objective function. We acknowledge an important point here, that the state of the Hessian during training says *nothing* about the *generalisation error*. We simply note that if we have a non-PD Hessian during the training, there will be no minimum to converge to *on the datapoint at which it was evaluated*, in which case training can only cause weight divergence. We would therefore like to know conditions under which the Hessian will be non-PD.

If the Hessian matrix is positive definite, then all elements on the leading diagonal are positive-valued; therefore if any element on that diagonal is zero or less, the entire matrix *cannot* be positive definite. Assume we have an estimator that is a linear combination of a number of nonlinear functions ϕ , so

$$f_i = \sum_{k=1}^K w_{ki}\phi_{ki}. \tag{36}$$

Examples of estimators in this class are Multi-Layer Perceptrons using linear output nodes, Polynomial Neural Networks, and Radial Basis Functions. Now, for an arbitrary Hessian diagonal element corresponding to the q th weight in the output layer of the i th network, w_{qi} , we can show (derivation given in Appendix A) that

$$\frac{\partial^2 e_i}{\partial w_{qi}^2} = \left[1 - \lambda\left(1 - \frac{1}{M}\right)\right]\phi_{qi}^2. \tag{37}$$

where in the case of RBF networks, ϕ_{qi}^2 is the squared output of the q th basis function in the i th network. If this element, Equation (37), equates to zero or less, the entire Hessian matrix *is guaranteed to be non-positive definite*. Therefore we would like the following inequality to hold,

$$\begin{aligned} 0 &< \left[1 - \lambda\left(1 - \frac{1}{M}\right)\right]\phi_{qi}^2 \\ 0 &< \phi_{qi}^2 - \lambda\phi_{qi}^2\left(\frac{M-1}{M}\right) \\ \lambda\phi_{qi}^2\left(\frac{M-1}{M}\right) &< \phi_{qi}^2 \\ \lambda &< \frac{\phi_{qi}^2}{\phi_{qi}^2\left(\frac{M-1}{M}\right)} \\ \lambda &< \frac{M}{M-1}. \end{aligned} \tag{38}$$

Since the effect of ϕ_{qi} cancels out,⁵ we find that this inequality is *independent* of all other network parameters, so it is *a constant for any ensemble architecture using estimators of this form*

5. We note that we assume a basis function $\phi_{qi} \neq 0$, to avoid divide by zero problems—this does not always hold, for example when using hyperbolic tangent activations; however here we assume either sigmoid activation or a Gaussian RBF.

and a simple average combination function. This defines an upper bound for λ and, since we know the relationship between the two strength parameters from Equation (30), we can also show a bound for γ ,

$$\lambda_{upper} = \frac{M}{M-1} \qquad \gamma_{upper} = \frac{M^2}{2(M-1)^2}. \tag{39}$$

When λ or γ is varied beyond these upper bounds, the Hessian matrix is guaranteed to be non-positive definite. Figure 2 plots λ_{upper} and the equivalent γ_{upper} for different ensemble sizes. We see that as the size increases, λ_{upper} asymptotes to 1, and γ_{upper} to 0.5. For larger ensembles, e.g. $M > 10$, this therefore lends concrete theoretical evidence to Liu’s proposed bound of $\lambda = 1$. However, for small M , the bound shows values larger than $\lambda = 1$ may still retain a positive definite Hessian matrix.

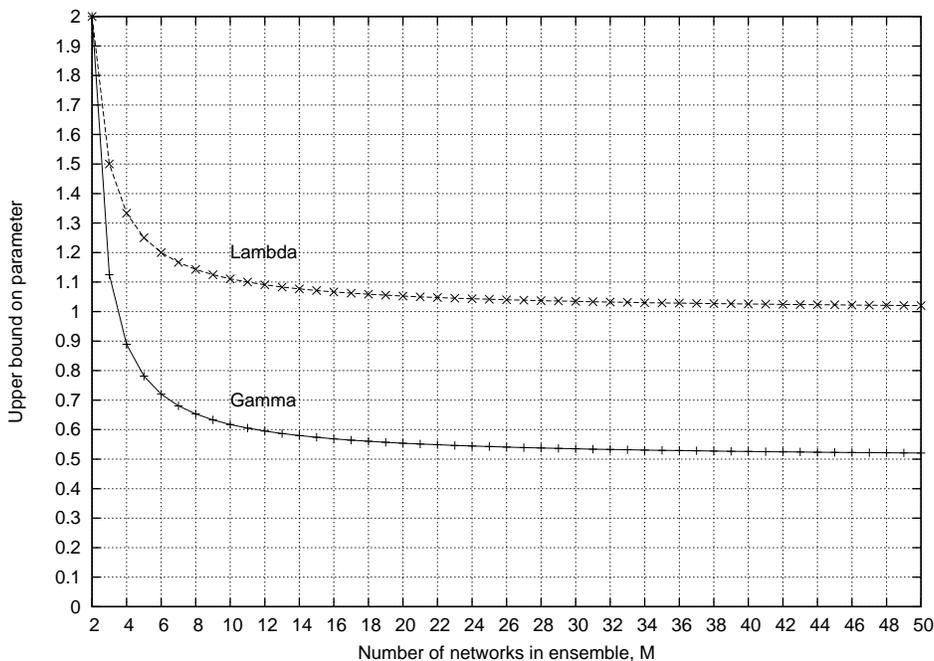


Figure 2: The Upper bound on γ and λ .

Our bound was determined on the premise that the leading diagonal containing negative elements implies a non-PD Hessian matrix. However, it could easily be the case that the leading diagonal is all positive, yet the entire matrix is still non-PD. This implies that the matrix could become non-PD *before* our upper bound is reached. Our bound is therefore a conservative one, and it may be possible to define a tighter bound. The question of whether a tighter bound can be defined can be phrased as “*Are there any general conditions for the off-diagonal elements of the Hessian, that will force non-positive definiteness, in spite of all leading diagonal elements being positive?*”. Any such analytical conditions based on the Hessian will almost certainly be input-dependent—the advantage of our bound is that it is a constant for a given ensemble, dependent only on the number of ensemble members. However, the utility of the bound depends entirely on how tight it is—using

a neural network ensemble it would be pointless if the weights diverged significantly *before* the bound is reached. To validate this hypothesis we now engage in empirical testing.

5. Empirically Validating the Proposed Bound

The purpose of this section is to determine how useful our theoretical upper bound can be in practice. We remind the reader again that our bound is not computed in reference to generalisation error, and we now wish to evaluate whether it can be practically useful in this context. When varying γ , if the network weights diverge significantly before the upper bound is reached, then the bound is not tight and therefore of little use. Alternatively, it could simply be that certain ensemble configurations do not show any benefit from using NC, in which case NC *itself* is of no use, and neither is our parameter bound. We now investigate these issues.

5.1 Data Sets

We use the Boston Housing data set, where the problem is to predict the median house price given a number of demographic features. There are 506 examples, each containing 13 input variables (12 continuous, 1 binary), and 1 continuous output variable in the range 0 to 50. All input variables were linearly rescaled, independently of each other, to be in the range $[0, 1]$, and the output variable was linearly rescaled to $[-1, 1]$. A five-fold cross validation procedure was used, so keeping 20% of the data as a holdout set, and using the remaining 80% for training and validation. With the Boston data set this equates to 304 for training, 101 for validation, and 101 for testing. The validation data was used to perform early stopping by the following procedure: train while noting the validation error every 50 epochs; if the validation error has risen in comparison to 500 epochs ago, terminate training and reset the weights to the best point within that 500 epoch window (at a resolution of 50 epochs) according to the validation data.

The second data set was generated (Friedman (1991)) by the function

$$h(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20 \left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5 + \eta, \quad (40)$$

where $\mathbf{x} = [x_1, \dots, x_{10}]$ is an input vector whose components are drawn uniformly at random from $[0, 1]$, and η is a noise component drawn from $N(0, 1)$, i.e. mean zero and variance 1.0. Totally there are 10 continuous valued attributes, but only the first 5 are used in the function, leaving the last 5 as irrelevant characteristics that the algorithm has to learn to ignore. We used a data set of size 1000, using the same five-fold cross validation procedure as described above.

The third data set used was the *LogP* data, recently used in (Tino et al. (2004)). This is a highly nonlinear pharmaceutical data set, where the task is to predict the *partition coefficient* of a chemical compound, allowing one to determine certain uptake properties of the molecule. The data set has 14 continuous input variables, and 1 continuous output variable in the range $[-4.2, +9.9]$. There are 6912 examples, which we used in the same cross-validation procedure as above.

For all three data sets, each ensemble was evaluated over the 5 data folds and over 30 trials of random weights, giving 150 trials for each run.

5.2 When Does the NC Technique Work Well?

Empirical analyses of NC have been shown on several other occasions and on several other data sets (Liu et al. (2000); Liu and Yao (1997); Brown (2004)). The point of this section is to characterise some general conditions of ensemble architecture under which NC seems to succeed in comparison to a simple ensemble.

We train several different ensemble architectures using a range of γ values (at a resolution of 0.05), the optimum γ value was located according to the validation data, and finally evaluated on the testing data. This was compared to using $\gamma = 0$, where it should be remembered that $\gamma = 0$ is exactly equivalent to simple ensemble learning, i.e. training each network independently of the others *without* NC learning. We first varied the number of networks in the ensemble, using a fixed individual network size of 6 hidden nodes. Figure 3 shows results for the Friedman data, figure 4 for Boston, and figure 5 for LogP; 95% confidence intervals are indicated. With the Friedman and Boston data sets, a general trend that can be noted is that larger relative gains seem to be made with larger ensemble sizes.

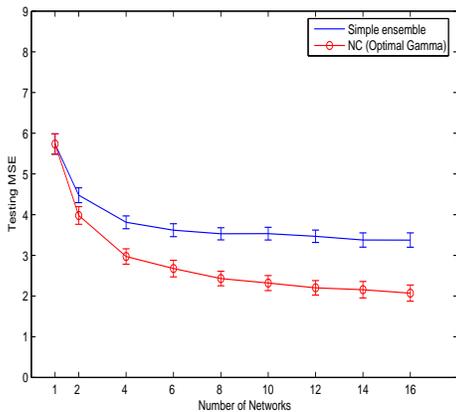


Figure 3: Friedman, $\gamma = 0$ versus optimal γ , 6 hidden nodes per network

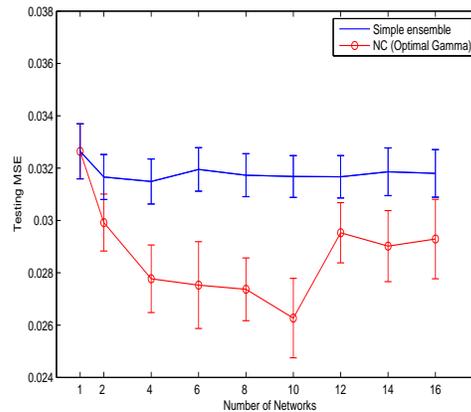


Figure 4: Boston, $\gamma = 0$ versus optimal γ , 6 hidden nodes per network

However, with the LogP data, relative performance does not seem to increase with the size of the ensemble. If we make the component networks much simpler, 2 hidden nodes as in figure 6, we see the same recognisable trends as in the other data sets. The general rule here, supported by previous empirical work on NC, seems to be to use very simple networks—in this case we can see that an ensemble of 16 networks, each with 2 hidden nodes, has equalled the performance of a similarly sized ensemble, using 6 hidden nodes per network.

Figures 7 and 8 show the gains as we vary the *complexity* (i.e. number of hidden nodes per network) of the individual ensemble members. We can note here that the gain from using NC *decreases* as we increase the complexity of the networks. Regarding again figures 3 to 6 these results indicate that NC is of most use when we have large ensembles of relatively low complexity ensemble members. This is emphasized further looking at figure 10, where we see that an ensemble of 6 networks using 2 hidden nodes, and using NC, can equal the performance of the same ensemble using

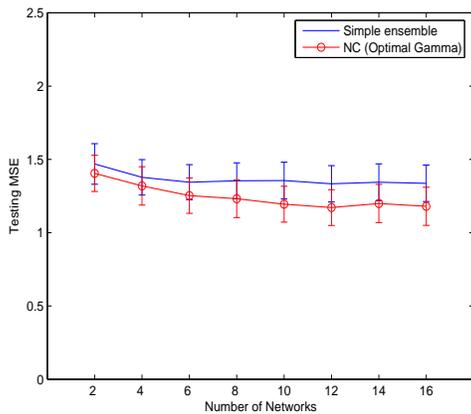


Figure 5: LogP, $\gamma = 0$ versus optimal γ , 6 hidden nodes per network

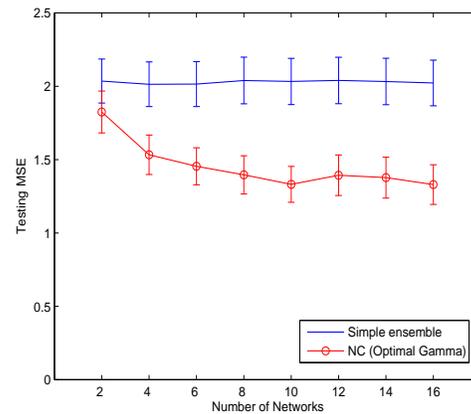


Figure 6: LogP, $\gamma = 0$ versus optimal γ , 2 hidden nodes per network

far more complex networks. Additionally, in these situations, when γ is set optimally, significantly faster convergence and lower generalisation error for a fixed number of epochs were observed.

5.3 How Tight is the Bound?

We now turn to examining the behaviour of the *generalization error* as we move γ toward its upper bound. Figures 11, 13 and 15 show the performance as γ is changed, with several different sizes of ensemble—each network has fixed complexity at 6 hidden nodes. A distinctive pattern is observed: a virtually monotonic decrease in error as we increase γ , up to a particular “threshold”, beyond which the error rises rapidly. On closer examination of the networks trained with these high γ values, it was observed that the network weights had diverged to excessively large values. The point at which divergence occurs seems to move downward as we increase the size of the ensemble. Figures 12, 14 and 16 show the behaviour with a fixed ensemble size, $M = 6$, as we vary the individual complexity between 2 and 12 hidden nodes. Here we see a distinction from the results varying ensemble size: the divergence point seems largely unaffected by the complexity of the networks.

Using these results as a guide, we searched the range of γ at the finer resolution of 0.01 to locate the divergence point. Figure 18 shows this, illustrating that divergence seems extremely *invariant* to the choice of data set. We superimpose the predicted upper bound and note that as the number of networks increases, the divergence point and the upper bound both asymptote to 0.5, confirming that our bound is tight. We have also superimposed $\gamma = \frac{M}{2(M-1)}$, corresponding to when $\lambda = 1$. Zooming in on part of the plot allows us to see that the $\lambda = 1$ original bound is obeyed in most instances, but not all. We acknowledge of course that the *exact* location of the divergence point is of little consequence; the real point we wish to locate is the *optimum* γ value, and see if it provides significant improvements relative to other ensemble techniques; we will explore this in the next section.

In conclusion to the ‘upper bound’ issue, we note that we have provided theoretical evidence that supports that $\lambda = 1$ bound in the case of large M , but the bound remains loose for small M , and

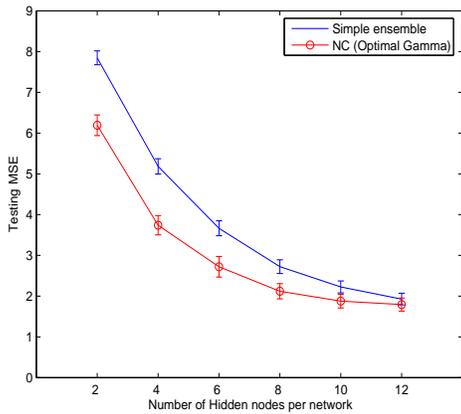


Figure 7: Friedman, $\gamma = 0$ versus optimal γ for 6 networks

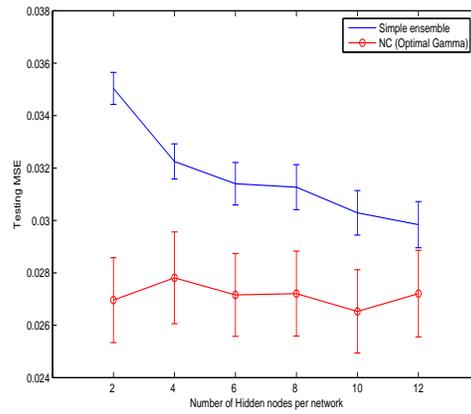


Figure 8: Boston, $\gamma = 0$ versus optimal γ for 6 networks

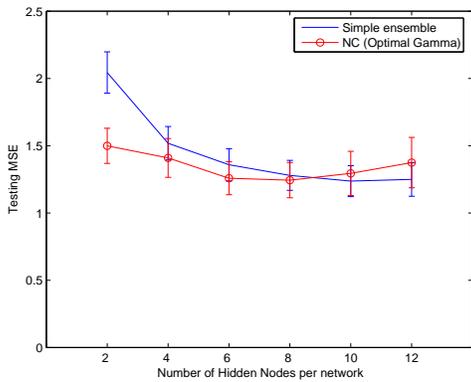


Figure 9: LogP, $\gamma = 0$ versus optimal γ for 6 networks

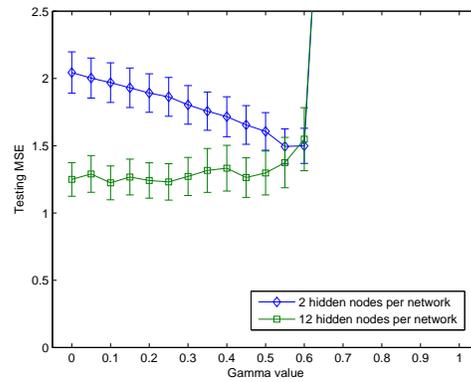


Figure 10: LogP, plotting testing MSE against Gamma for an ensemble of 6 networks

$\lambda = 1$ (or equivalently $\gamma = \frac{M}{2(M-1)}$) seems to be a useful heuristic bound. A possible justification for this is to remember that as we approach $\lambda = 1$, we treat the ensemble more and more as a single learning unit—beyond this we would be introducing a greater emphasis on covariance than is in the overall ensemble objective function; whether this bound can be strictly proved remains an open question.

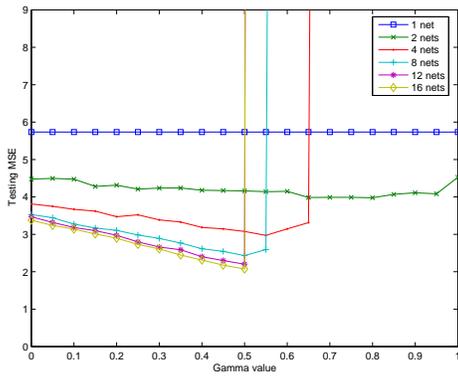


Figure 11: Friedman, varying γ with 6 hidden nodes per network

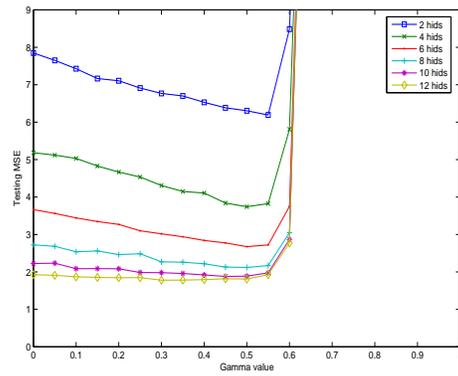


Figure 12: Friedman, varying γ with 6 networks

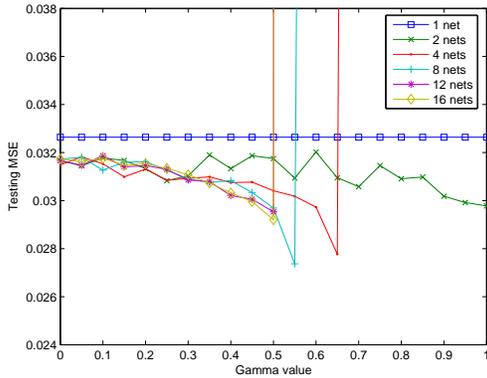


Figure 13: Boston, varying γ with 6 hidden nodes per network

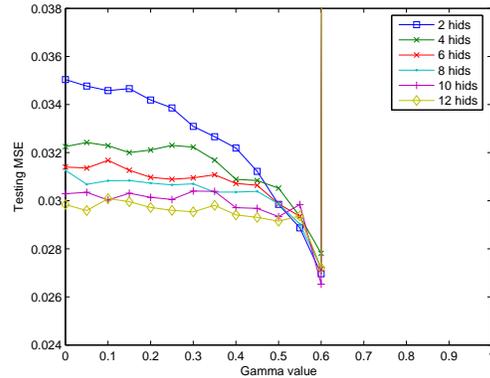


Figure 14: Boston, varying γ with 6 networks

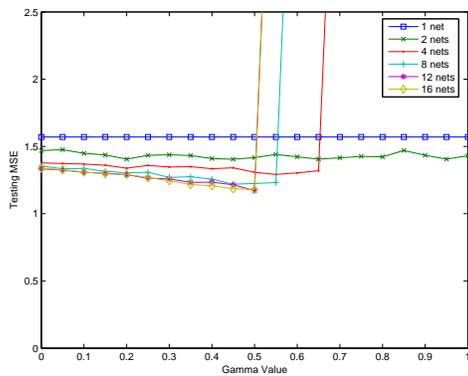


Figure 15: LogP, varying γ with 6 hidden nodes per network

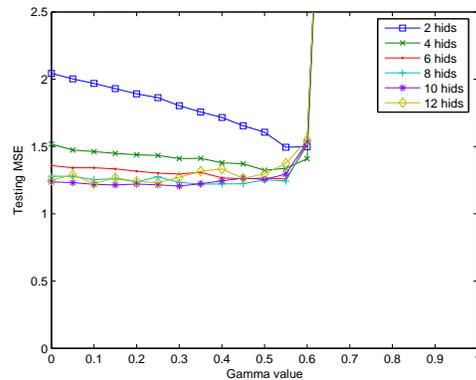


Figure 16: LogP, varying γ with 6 networks

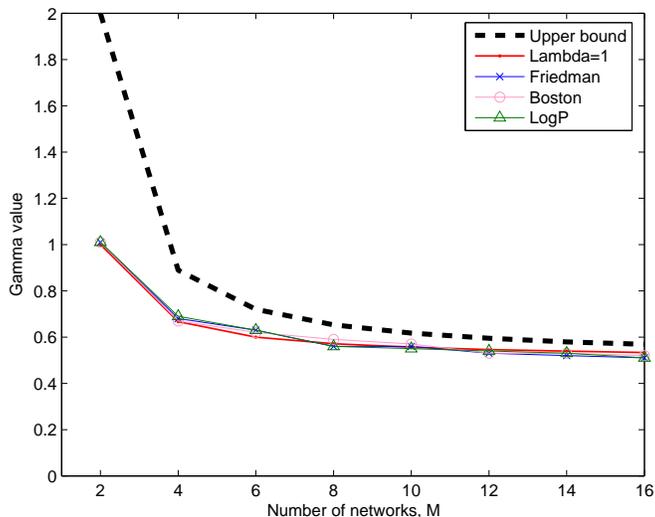


Figure 17: γ -value at which divergence of weights was observed for the data sets.

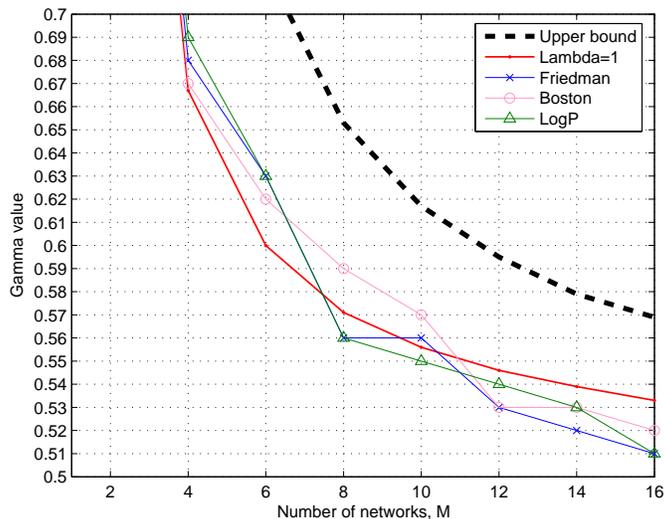


Figure 18: γ -value at which divergence occurred, zoomed in.

6. Further Empirical Comparisons

In this section we will compare the NC framework to other competitive ensemble approaches, using MLPs as the base estimator. Additionally we will illustrate that the NC framework is indeed general, by using RBF regressors, and showing that very similar empirical patterns emerge, obeying our upper bound for the γ parameter.

6.1 Comparing NC to Other Popular Ensemble Approaches

Performing valid empirical comparisons with existing works in the literature is a notoriously difficult task; slight differences in experimental setup can easily invalidate the procedure. In particular, training/testing data must be exactly the same between two systems to be compared. The LogP data has been used in previous work (Tino et al. (2004)) by one of the current authors—we obtained the exact data split used in that work to test NC against their results. Of the 6912 examples, 5530 were used for training, 691 for validation and 691 for testing. We used 12 networks, each with 6 hidden nodes. Using the validation data, the optimum $\gamma = 0.5$ was determined. Results in table 2 show how NC compares with other state-of-the-art techniques; 95% confidence intervals are indicated where available. As an additional useful statistic, Tino et al. (2004) computed the ION (Improvement over Naive) value. This is the percentage improvement relative to a naive predictor (with an MSE of 2.69) which predicts a constant for any input, equal to the mean target value in the training data. The best achievable improvement in their experiments was 77.7%, the Gaussian Process learner, while here we see NC achieves 78.2%.

<i>System</i>	<i>Testing MSE (conf)</i>	<i>ION %</i>
NC, 12 MLPs, $\gamma = 0.5$	0.5866(± 0.0168)	78.2%
Gaussian Processes	0.601	77.7%
Hierarchical Mixture of Experts	0.658	75.5%
Simple ensemble, 12 MLPs	0.7692(± 0.0154)	71.4%

Table 1: Comparing NC to other state-of-the-art learning techniques on the *LogP* data.

To further empirically verify NC, we now compare it to two other popular ensemble techniques, Adaboost.R2 and bagging. Figures 19 to 22 show results, again following the empirical procedures described in section 5 - all Boosted and Bagged networks were trained with early stopping. We note that on the Friedman data, NC significantly outperforms both boosting and bagging, increasing its lead as the ensemble size is increased. The Boston data shows that NC is obviously not a panacea technique - boosting and bagging significantly outperform it in this situation. From this and previous experiments with NC, we hypothesize that the noisy nature of the Friedman data is ideally suited to the flexibility allowed by NC's γ parameter, explicitly varying the *fit of the ensemble model* to the data as needed, whereas boosting and bagging do not have this extra free parameter. We note that a full empirical benchmarking of NC and its behaviour with noisy data is underway, but outside the scope of this article.

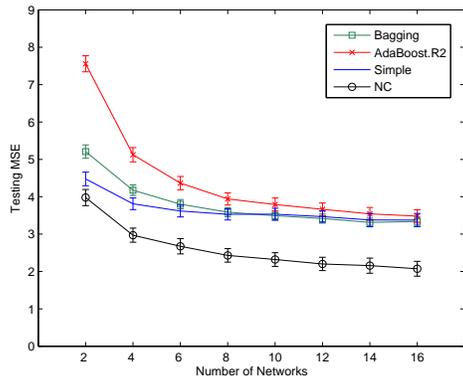


Figure 19: Friedman, varying number of networks (6 hidden nodes in each)

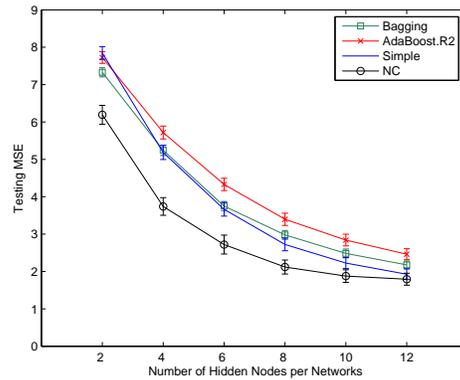


Figure 20: Friedman, varying number of hidden nodes per network (ensemble of 6 networks)

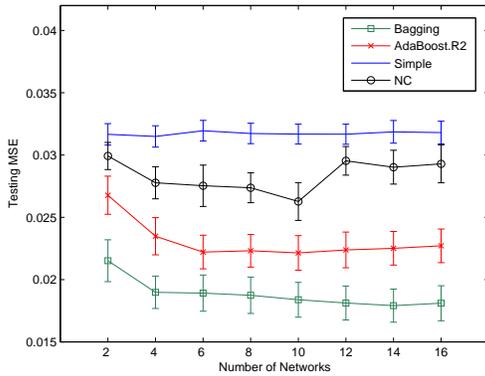


Figure 21: Boston, varying number of networks (6 hidden nodes in each)

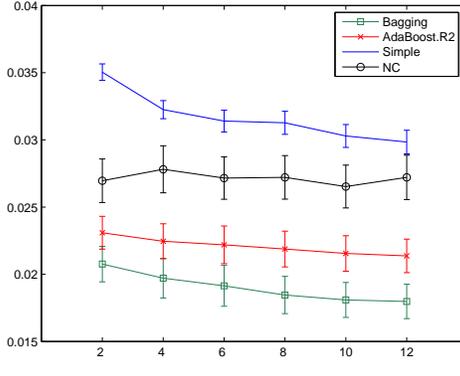


Figure 22: Boston, varying number of hidden nodes per network (ensemble of 6 networks)

6.2 Using NC with an Ensemble of RBF Networks

We now briefly illustrate that NC can be applied to regression estimators of other types, not just multi-layer perceptrons. We use an ensemble of Radial Basis Function networks, using Gaussian basis functions. Centres and widths are initialised randomly, then a full gradient descent is performed on all parameters, using the NC penalty framework as previously described. Table 2 shows that an ensemble of RBF networks each with 50 centres can outperform an MLP ensemble each with 50 sigmoidal hidden nodes, and applying NC to the RBF ensemble allows further gain. Finally in figure 23 we see the effect of varying γ on both the MLP and RBF ensemble. As previously observed, with very complex individuals NC cannot provide further error reduction. Here we note two points. Firstly, though an MLP ensemble cannot benefit, an RBF ensemble of the same size *can*

benefit from NC. Secondly, and most importantly, we see it again obeys our predicted upper bound on γ .

System	Testing MSE (conf)
RBF: 5 x 50 basis functions, NC $\gamma = 0.5$	0.0229 (± 0.001)
RBF: 5 x 50 basis functions	0.0263(± 0.001)
MLP: 5 x 50 hidden nodes, NC $\gamma = 0.5$	0.0313(± 0.001)
MLP: 5 x 50 hidden nodes	0.0319(± 0.001)

Table 2: Using NC with an ensemble of RBF networks on the Boston data set

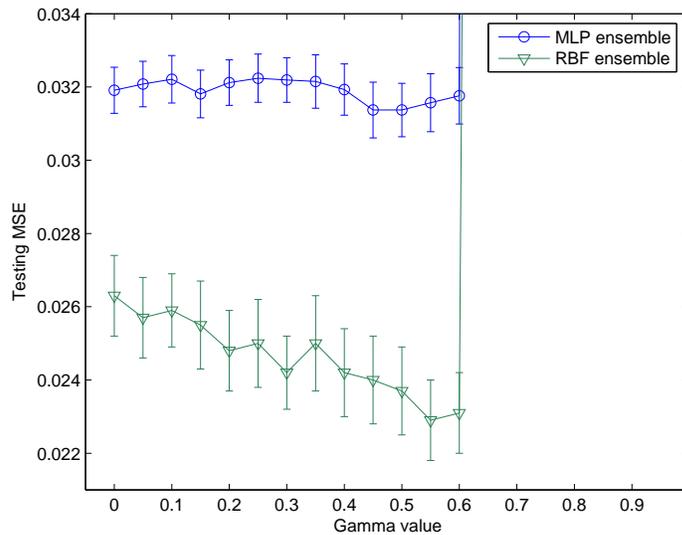


Figure 23: LogP data: The effect of varying the NC γ parameter on an RBF and MLP ensemble of size $M = 5$, noting that our predicted upper bound $\gamma_{upper} = 0.78125$ holds for the RBF ensemble.

7. Conclusions

We have investigated the issue of how to explicitly *manage* the correlations in an ensemble of regression estimators. We made important observations on the relationships between the Ambiguity decomposition (Krogh and Vedelsby (1995)) and the bias-variance-covariance decomposition (Ueda and Nakano (1996)). From this base, we provided a thorough critique of negative correlation (NC) learning (Liu (1998)), a technique that extended from Rosen (1996), and developed in the evolu-

tionary computation literature. We showed that using a penalty term and coefficient, NC *explicitly includes the covariance portion of the ensemble MSE in its error function*. This article has served to illustrate that NC is not merely a heuristic technique, but *fundamentally* tied to the dynamics of training an ensemble system with the mean squared error function. The observations we made are in fact all *properties of the mean squared error function*. NC is therefore best viewed as a *framework* rather than as an algorithm. The NC framework can be applied to ensembles of *any nonlinear regression estimator* combined in an ensemble \bar{f} of the form

$$\bar{f}(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}). \quad (41)$$

In addition, an upper bound on the strength parameter was shown to apply when each estimator f_i is of the form

$$f_i(\mathbf{x}) = \sum_{k=1}^K w_{ki} \phi_{ki}(\mathbf{x}). \quad (42)$$

Examples of estimators in this class are multi-layer perceptrons with linear output nodes, and Radial Basis Function networks, indeed any estimator that can be cast in a generalised linear regression framework. To derive the bound, we observed that positive definiteness of the Hessian matrix can be determined by checking just the first K leading diagonal elements. We verified this bound empirically, and although the bound is tight for larger ensembles, it remains loose for size $M < 10$, and a useful empirical bound of $\gamma = \frac{M}{2(M-1)}$ seems to apply. These results seem to suggest a general set of guidelines for application of the NC framework. The common trend was to see increasing utility of NC with larger ensembles of relatively low individual complexity, with optimum γ tending to 0.5. We therefore recommend a starting point as: ensemble size $M \geq 10$, number of hidden nodes between 2 and 5, and a penalty strength parameter of $\gamma = 0.5$. This will of course be problem dependent, most significantly the number of hidden nodes—what is ‘low complexity’ for one task will not be for another—but we believe it does provide good general guidance. In addition, it seems sensible from our investigations that some sort of annealing of the parameter during the learning process, from zero up towards the bound, may show further performance benefits.

We then engaged in a detailed study of the error gradient and how it changes when using NC learning. We showed that the error of an NC learning ensemble can be broken down into four components, each with its own interpretation with respect to the current state of the ensemble. Further to this we noted that NC allows a smooth transition of the error gradients between that of a fully parallel ensemble system and a single estimator. This raises a point on the nature of *overfitting* in ensembles. It is well known that overfitting of the individual estimators can be beneficial in an ensemble system (Sollich and Krogh (1996)), but obviously overfitting the entire ensemble as a unit is an undesirable prospect. With this new information about NC, *what* should we overfit?

Appendix A. Calculations Supporting the Strength Parameter Bound

We now present additional calculations supporting the work on the upper bound for the λ and γ parameters, as in Section 4.3. Assuming an estimator which is a linear combination of other functions ϕ , we wish to derive one of the entries in the leading diagonal of the Hessian matrix. The diagonal element corresponding to the q th weight in the i th estimator is $\frac{\partial^2 e_i}{\partial w_{qi}^2}$. If this is zero or less, then the Hessian is guaranteed to be non-positive definite, an undesirable prospect. Making use of

the product rule, we have

$$\begin{aligned}\frac{\partial e_i}{\partial w_{qi}} &= \frac{\partial e_i}{\partial f_i} \frac{\partial f_i}{\partial w_{qi}} \\ \frac{\partial^2 e_i}{\partial w_{qi}^2} &= \left[\frac{\partial}{\partial w_{qi}} \frac{\partial e_i}{\partial f_i} \right] \frac{\partial f_i}{\partial w_{qi}} + \left[\frac{\partial}{\partial w_{qi}} \frac{\partial f_i}{\partial w_{qi}} \right] \frac{\partial e_i}{\partial f_i}.\end{aligned}\quad (43)$$

Taking the first term on the right hand side,

$$\begin{aligned}\frac{\partial e_i}{\partial f_i} &= (f_i - t) - \lambda(f_i - \bar{f}) \\ \frac{\partial}{\partial w_{qi}} \frac{\partial e_i}{\partial f_i} &= \phi_{qi} - \lambda(\phi_{qi} - \frac{1}{M}\phi_{qi}) \\ &= \left(1 - \lambda\left(1 - \frac{1}{M}\right)\right)\phi_{qi}.\end{aligned}\quad (44)$$

Now for the second term, remembering eq (42), we have

$$\frac{\partial f_i}{\partial w_{qi}} = \phi_{qi} \quad (45)$$

$$\frac{\partial}{\partial w_{qi}} \frac{\partial f_i}{\partial w_{qi}} = \frac{\partial^2 f_i}{\partial w_{qi}^2} = 0. \quad (46)$$

Therefore we simply have

$$\frac{\partial^2 e_i}{\partial w_{qi}^2} = \left[\left(1 - \lambda\left(1 - \frac{1}{M}\right)\right)\phi_{qi} \right] \phi_{qi} + [0] \frac{\partial e_i}{\partial f_i} \quad (47)$$

$$= \left(1 - \lambda\left(1 - \frac{1}{M}\right)\right)\phi_{qi}^2. \quad (48)$$

It is interesting to observe that since we have

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) - \lambda(f_i - \bar{f}) \quad (49)$$

$$\frac{\partial^2 e_i}{\partial f_i^2} = 1 - \lambda\left(1 - \frac{1}{M}\right). \quad (50)$$

then we can see

$$\frac{\partial^2 e_i}{\partial w_{qi}^2} = \frac{\partial^2 e_i}{\partial f_i^2} \phi_{qi}^2. \quad (51)$$

This demonstrates that, since ϕ_{qi}^2 is positive, the sign of the leading diagonal entry $\frac{\partial^2 e_i}{\partial w_{qi}^2}$ in the Hessian is decided by the sign of $\frac{\partial^2 e_i}{\partial f_i^2}$.

Appendix B. The Relationship between Ambiguity and Covariance

We now show the exact link between the Ambiguity decomposition and the bias-variance-covariance decomposition. The bias-variance-covariance decomposition gives us

$$E\{(\bar{f} - t)^2\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1 - \frac{1}{M}\right)\overline{covar}. \quad (52)$$

Now using the Ambiguity decomposition, we have the result

$$E\left\{\frac{1}{M}\sum_i (f_i - t)^2 - \frac{1}{M}\sum_i (f_i - \bar{f})^2\right\} = \overline{bias}^2 + \frac{1}{M}\overline{var} + \left(1 - \frac{1}{M}\right)\overline{covar}. \quad (53)$$

It would be interesting to understand what portions of the bias-variance-covariance decomposition correspond to the ambiguity term. We place an α in front of the Ambiguity term, then derive the relationship between the left and right sides of equation (53). Wherever the α appears in the derivation will indicate how the Ambiguity term plays a role in the bias-variance-covariance decomposition. We have

$$\begin{aligned} e_{ens} &= E\left\{\frac{1}{M}\sum_i [(f_i - t)^2 - \alpha(f_i - \bar{f})^2]\right\} \\ &= \frac{1}{M}\sum_i \left[E\left\{(f_i - E\{\bar{f}\} + E\{\bar{f}\} - t)^2 - \alpha(f_i - E\{\bar{f}\} + E\{\bar{f}\} - \bar{f})^2\right\}\right]. \end{aligned}$$

now multiply out the brackets, thus

$$\begin{aligned} e_{ens} &= \frac{1}{M}\sum_i \left[E\left\{(f_i - E\{\bar{f}\})^2 + (E\{\bar{f}\} - t)^2 + 2(f_i - E\{\bar{f}\})(E\{\bar{f}\} - t) \right. \right. \\ &\quad \left. \left. - \alpha(f_i - E\{\bar{f}\})^2 - \alpha(E\{\bar{f}\} - \bar{f})^2 - 2\alpha(f_i - E\{\bar{f}\})(E\{\bar{f}\} - \bar{f})\right\}\right]. \end{aligned}$$

and evaluate the expectation and summation, giving us

$$\begin{aligned} e_{ens} &= \frac{1}{M}\sum_i E\left\{(f_i - E\{\bar{f}\})^2\right\} + (E\{\bar{f}\} - t)^2 \\ &\quad - \alpha\frac{1}{M}\sum_i E\left\{(f_i - E\{\bar{f}\})^2\right\} - \alpha E\left\{(\bar{f} - E\{\bar{f}\})^2\right\} - 2\alpha E\left\{(\bar{f} - E\{\bar{f}\})(E\{\bar{f}\} - \bar{f})\right\}. \end{aligned}$$

and finally by rearranging the last term we obtain

$$\begin{aligned} e_{ens} &= \frac{1}{M}\sum_i E\left\{(f_i - E\{\bar{f}\})^2\right\} + (E\{\bar{f}\} - t)^2 \\ &\quad - \alpha\frac{1}{M}\sum_i E\left\{(f_i - E\{\bar{f}\})^2\right\} + \alpha E\left\{(\bar{f} - E\{\bar{f}\})^2\right\}. \end{aligned}$$

Obviously now if we remove the α term that we have been using, this would simplify to give us the squared bias of \bar{f} , plus the variance of \bar{f} : which we could then break down further using the bias-variance-covariance decomposition as we showed earlier. The interesting part here though, is

the term that would cancel out. The expected value of the Ambiguity term is equal to whatever parts of this that contain the α term. Therefore,

$$\begin{aligned} E\left\{\frac{1}{M} \sum_i (f_i - \bar{f})^2\right\} &= \frac{1}{M} \sum_i E\{(f_i - E\{\bar{f}\})^2\} - E\{(\bar{f} - E\{\bar{f}\})^2\} \\ &= \Omega - \text{var}(\bar{f}). \end{aligned}$$

And the other side of the Ambiguity decomposition, the expected value of the average individual error is whatever parts *do not* contain α , this is

$$\begin{aligned} E\left\{\frac{1}{M} \sum_i (f_i - t)^2\right\} &= \frac{1}{M} \sum_i E\{(f_i - E\{\bar{f}\})^2\} + (E\{\bar{f}\} - t)^2 \\ &= \Omega + \text{bias}(\bar{f})^2. \end{aligned}$$

This interaction term, Ω , is present in both sides, and cancels out to allow the normal bias-variance decomposition of ensemble error. But what does it *mean*? If we examine it a little further, we see

$$\begin{aligned} \frac{1}{M} \sum_i E\{(f_i - E\{\bar{f}\})^2\} &= \frac{1}{M} \sum_i E\{(f_i - E\{f_i\} + E\{f_i\} - E\{\bar{f}\})^2\} \\ &= \frac{1}{M} \sum_i E\{(f_i - E\{f_i\})^2\} + \frac{1}{M} \sum_i (E\{f_i\} - E\{\bar{f}\})^2. \end{aligned}$$

where we have used that $E\{f_i E\{f_i\}\} = E\{f_i\}^2$ and also $E\{f_i E\{\bar{f}\}\} = E\{f_i\} E\{\bar{f}\}$. This shows that the interaction term, Ω , is the average variance of the estimators, plus the average squared deviation of the expectations of the individuals from the expectation of the ensemble.

Appendix C. Further Gradient Analysis of NC Learning

We have seen that the MSE of an ensemble system can be interpreted in two ways: firstly with the Ambiguity decomposition, and secondly with the bias-variance-covariance decomposition. We now present a third way to understand the dynamics of a regression ensemble, in reference to the gradient of the error function. Regard the architecture in figure 24. This is an ensemble of three

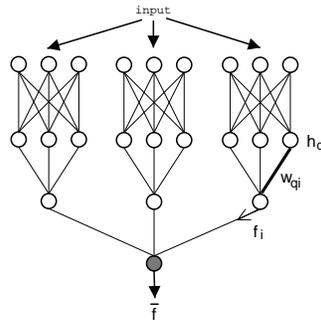


Figure 24: A typical ensemble architecture

MLPs, with three inputs and three hidden nodes each, using a uniformly weighted combination as the ensemble output. We desire to update the weight, w_{qi} , marked in bold—this is one of the output layer weights for the i th network (connected to the q th hidden node). If we consider the ensemble as a single entity, then the error of this system at a single point is defined as

$$e_{ens} = \frac{1}{2}(\bar{f} - t)^2. \quad (54)$$

In this case, an update to the weight w_{qi} would involve the gradient

$$\begin{aligned} \frac{\partial e_{ens}}{\partial w_{qi}} &= \frac{\partial e_{ens}}{\partial f_i} \frac{\partial f_i}{\partial w_{qi}} \\ &= \frac{1}{M}(\bar{f} - d) \frac{\partial f_i}{\partial w_{qi}}. \end{aligned} \quad (55)$$

Note that we are assuming an ensemble of networks with linear output functions—if this is the case, the second term in the above error gradient, $\frac{\partial f_i}{\partial w_{qi}}$ evaluates to simply the output of the relevant hidden node. The error gradient is therefore proportional to $\frac{\partial e_{ens}}{\partial f_i}$, and we can simplify our calculations below by omitting the reference to the hidden node since it just acts as a scaling component.

We calculated (55) in one simple step using the chain rule, treating the ensemble as a single unit—if we perform this instead starting from the decomposed form of the ensemble error, it highlights more interesting results. We use the Ambiguity decomposition, and additionally break the error into two components, where the first term concerns estimator i , and the second concerns all the other estimators $j \neq i$,

$$\begin{aligned} \frac{1}{2}(\bar{f} - t)^2 &= \frac{1}{M} \sum_i \left[\frac{1}{2}(f_i - t)^2 - \frac{1}{2}(f_i - \bar{f})^2 \right] \\ &= \frac{1}{M} \left[\frac{1}{2}(f_i - t)^2 - \frac{1}{2}(f_i - \bar{f})^2 \right] \\ &\quad + \frac{1}{M} \sum_{j \neq i} \left[\frac{1}{2}(f_j - t)^2 - \frac{1}{2}(f_j - \bar{f})^2 \right]. \end{aligned} \quad (56)$$

If we do this we discover that the gradient of the ensemble error function is a sum of four distinct components, shown and described in table 3. Each of these components contributes to the gradient of the ensemble error in eq. (55). If we take the $\frac{1}{M}$ on the outside and label the components, we can make an interesting observation.

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M} \left[\underbrace{(f_i - t)}_A - \underbrace{(f_i - \bar{f})}_B + \underbrace{\frac{1}{M}(f_i - \bar{f})}_C + \underbrace{\frac{1}{M} \sum_{j \neq i} (f_j - \bar{f})}_D \right] \quad (57)$$

We now see that the gradient of the individual, and the gradient of the ensemble as a single unit, can be expressed as combinations of these components; thus we have

$$\frac{\partial e_i}{\partial f_i} = (f_i - t) = A \quad (58)$$

<i>Component</i>	<i>Interpretation</i>
$\frac{1}{M}(f_i - t)$	This is the component of the error gradient due to the difference between the i th network output and the desired output (due to the fact that f_i is changing.)
$-\frac{1}{M}(f_i - \bar{f})$	This is the component of the error gradient due to the difference between f_i and \bar{f} (due to the fact that f_i is changing.)
$\frac{1}{M^2}(f_i - \bar{f})$	This is the component of the error gradient due to the difference between f_i and \bar{f} (due to the fact that \bar{f} changes, because f_i is changing.)
$\frac{1}{M^2} \sum_{j \neq i} (f_j - \bar{f})$	This is the component of the error gradient due to the differences between the f_j s and \bar{f} (due to the fact that \bar{f} changes, because f_i is changing.)

Table 3: Ensemble gradient components

$$\frac{\partial e_{ens}}{\partial f_i} = \frac{1}{M}(\bar{f} - t) = \frac{1}{M}(A - B). \quad (59)$$

Furthermore, a simple rearrangement now shows the error gradient for f_i in an ensemble using NC is

$$\begin{aligned} \frac{\partial}{\partial f_i} \left[\frac{1}{2}(f_i - t)^2 - \gamma(f_i - \bar{f})^2 \right] &= (f_i - t) - 2\gamma \left(1 - \frac{1}{M}\right)(f_i - \bar{f}) \\ &= (f_i - t) - 2\gamma \left[(f_i - \bar{f}) - \frac{1}{M}(f_i - \bar{f}) \right] \\ &= A - 2\gamma(B - C). \end{aligned} \quad (60)$$

Alternatively, because we know $\lambda = 2\gamma(1 - \frac{1}{M})$, this can also be expressed as $A - \lambda B$. From all this we can understand, a single framework, the relationships between minimising the simple ensemble error, the NC ensemble error, and a single network, described in table 4.

If we set $\lambda = 1$, or equivalently $\gamma = \frac{M}{2(M-1)}$, we see that the gradient of the individual error with NC is directly proportional to the gradient for the ensemble seen as a single entity, i.e.

$$\frac{\partial e_{ens}}{\partial w_{qi}} = \frac{1}{M} \frac{\partial e_i}{\partial w_{qi}}. \quad (61)$$

An alternative way of thinking about this is that all the minima are in the same locations, but the landscape is M times shallower—the effect of which could be duplicated with a smaller learning rate in the update rule. When we change γ within a certain range, we scale smoothly between the gradient of a single large entity, and that of a set of independently trained networks. The choice

<i>Algorithm</i>	<i>Components in error gradient for network i</i>
Simple Ensemble	A
Ensemble with NC	$A - 2\gamma(B - C)$ or $(A - \lambda B)$
Ensemble with NC, $\lambda = 1$ or equivalently $\gamma = \frac{M}{2(M-1)}$	$A - B$
Single large network (with fixed output layer weights)	$\frac{1}{M}(A - B)$

Table 4: Components of the Ensemble Error Gradient under different Algorithms

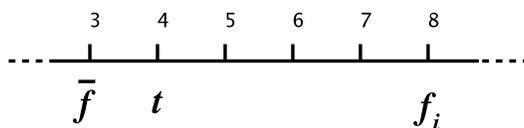


Figure 25: An regression example to illustrate how NC affects the error gradient.

of γ is problem-dependent, and it emerges that we can also understand the *optimal* setting of the parameter in this gradient-based context—consider the scenario in figure 25.

On a single datapoint, the network f_i is estimating too high at 8.0, which is right of the target $t = 4$. We have an ensemble of $M = 5$ networks, but for clarity the outputs of the other ensemble members are not shown; the resulting ensemble output is $\bar{f} = 3$, too low, left of the target. When updating the value of f_i , a simple ensemble will use the gradient measurement $(f_i - t) = 4$, resulting in f_i being shifted left, towards the target. However, this will cause the ensemble output \bar{f} to also shift *left*, moving *away from the target*. An ensemble using NC will include three gradient components,

$$\begin{aligned}
 A - 2\gamma(B - C) &= (f_i - t) - 2\gamma\left[(f_i - \bar{f}) - \frac{1}{M}(f_i - \bar{f})\right] & (62) \\
 &= 4 - 2\gamma\left(5 - \frac{1}{5}5\right) \\
 &= 4 - \gamma 8.
 \end{aligned}$$

If we choose $\gamma = 0.4$, this sum evaluates to 0.8, still a positive gradient for f_i , meaning the ensemble output will still be moved away from the target. If however we choose $\gamma = 0.6$, it evaluates to -0.8 , giving a pressure for the network f_i to move *away* from the target, causing the ensemble output to move *closer* to the target. The setting of the γ value provides a way of finding a trade-off

between these gradient components that will cause the ensemble output \bar{f} to move toward the target value t . This is obviously a purely hypothetical situation, and finding the optimal γ that allows this correct trade-off will be more difficult.

References

- G. Brown. *Diversity in Neural Network Ensembles*. PhD thesis, School of Computer Science, University of Birmingham, 2004.
- G. Brown, J. L. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. *Journal of Information Fusion*, 6(1):5–20, 2005a.
- G. Brown, J. L. Wyatt, and P. Sun. Between two extremes: Examining decompositions of the ensemble objective function. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 3541)*, Monterey, California, 2005b. Springer.
- J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19:1–141, 1991.
- G. Fumera and F. Roli. Linear combiners for classifier fusion: Some theoretical and experimental results. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2709)*, pages 74–83, Guildford, Surrey, 2003. Springer.
- S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
- J. V. Hansen. *Combining Predictors: Meta Machine Learning Methods and Bias/Variance and Ambiguity Decompositions*. PhD thesis, Aarhus Universitet, Datalogisk Institut, 2000.
- A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. *NIPS*, 7:231–238, 1995.
- L. I. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles. *Machine Learning*, (51):181–207, 2003.
- Y. Liu. *Negative Correlation Learning and Evolutionary Neural Network Ensembles*. PhD thesis, University College, The University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 1998.
- Y. Liu and X. Yao. Negatively correlated neural networks can produce best ensembles. *Australian Journal of Intelligent Information Processing Systems*, 4(3/4):176–185, 1997.
- Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4), 2000.
- H. Markowitz. Portfolio selection. *Journal of Finance*, 7, 1952.
- R. McKay and H. Abbass. Analyzing anticorrelation in ensemble learning. In *Proceedings of 2001 Conference on Artificial Neural Networks and Expert Systems*, pages 22–27, Otago, New Zealand, 2001.

- M. P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*. PhD thesis, Brown University, Institute for Brain and Neural Systems, 1993.
- B. E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science - Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 and 4):373–384, 1996.
- P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. 8:190–196, 1996.
- P. Tino, I. Nabney, B. S. Williams, J. Losel, and Y. Sun. Non-linear prediction of quantitative structure-activity relationships. *Journal of Chemical Information and Computer Sciences*, 44(5): 1647–1653, 2004.
- K. Tumer and J. Ghosh. Theoretical foundations of linear and order statistics combiners for neural pattern classifiers. Technical Report TR-95-02-98, Computer and Vision Research Center, University of Texas, Austin, 1995.
- N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks*, pages 90–95, 1996.
- X. Yao, M. Fischer, and G. Brown. Neural network ensembles and their application to traffic flow prediction in telecommunications networks. In *Proceedings of International Joint Conference on Neural Networks*, pages 693–698. IEEE Press, 2001. Washington DC.

Active Coevolutionary Learning of Deterministic Finite Automata

Josh Bongard

Hod Lipson

Computational Synthesis Laboratory

Sibley School of Mechanical and Aerospace Engineering

Ithaca, NY 14853, USA

JOSH.BONGARD@CORNELL.EDU

HOD.LIPSON@CORNELL.EDU

Editor: Stefan Wrobel

Abstract

This paper describes an active learning approach to the problem of grammatical inference, specifically the inference of deterministic finite automata (DFAs). We refer to the algorithm as the estimation-exploration algorithm (EEA). This approach differs from previous passive and active learning approaches to grammatical inference in that training data is actively proposed by the algorithm, rather than passively receiving training data from some external teacher. Here we show that this algorithm outperforms one version of the most powerful set of algorithms for grammatical inference, evidence driven state merging (EDSM), on randomly-generated DFAs. The performance increase is due to the fact that the EDSM algorithm only works well for DFAs with specific balances (percentage of positive labelings), while the EEA is more consistent over a wider range of balances. Based on this finding we propose a more general method for generating DFAs to be used in the development of future grammatical inference algorithms.

Keywords: grammatical inference, evolutionary computation, deterministic finite automata, active learning, system identification

1. Introduction

Grammatical inference is a popular machine learning domain (refer to Cicchello and Kremer, 2003, for an overview): it has wide applicability in both computational linguistics and related fields, as well as giving rise to a host of benchmark problems (Tomita, 1982; Lang et al., 1998) and competitions. Grammatical inference is a special case of the larger problem domain of inductive learning (Bergadano and Gunetti, 1995), which aims to construct models of some underlying system based on sets of positive and negative classifications. In one class of grammatical inference methods, the system is considered to be some kind of language or classifier, and models are represented as deterministic finite automata (DFA). Both the target system and models take strings of symbols as input (sentences), and produce binary classification as output (labellings), indicating whether that sentence belongs to the language or not. The problem of grammatical inference can also be considered a special instance of the problem of system identification (Ljung, 1999), in which some target system is inferred based solely on input/output data.

Grammatical inference methods that employ DFAs as models can be divided into two broad classes: passive and active learning methods. In passive methods, a set of training data is supplied to the algorithm for model construction. In active learning approaches, the algorithm has some

influence over which training data is labeled by the target DFA for model construction. Active learning approaches are typically iterative, in which membership queries are proposed periodically, often in response to some deficiency in the currently constructed models. In these iterative active approaches the amount of training data available for inference grows over time, unlike passive approaches, in which a fixed set of training data is used for model construction.

Passive methods usually make some assumption about the training data: a set of labeled training data is either generated by some auxiliary method randomly, or according to some predefined distribution. For example Pitt (1989), Porat and Feldman (1991), Dupont (1996) and Lang et al. (1998) assume a randomly-selected set of sample data; Luke et al. (1999) and Lucas and Reynolds (2005) assume equal amounts of positive and negative training data when inferring the Tomita languages (Tomita, 1982) by using the same training sets as previous researchers; Pao and Carr (1978) and Parekh and Honavar (1996) assume a structurally complete set; Oncina and Garcíá (1992) assume a characteristic sample; and Angluin (1981) assumes a live complete set. Once the sample data has been generated and labeled, inference is then conducted.

With the exception of randomly-generated training data, it is assumed that the training data is collected using some knowledge of the target system to be inferred. For example one necessary criterion for a structurally complete set of training data is that it covers every state transition of a DFA¹ (Pao and Carr, 1978; Parekh and Honavar, 1993; Dupont et al., 1994). This requires that the algorithm which generates the training data knows something about the structure of the DFA, namely its state transitions. This is advantageous as then it is possible to make performance guarantees regarding an inference algorithm working on that training data. However, for real-world usage of grammatical inference algorithms, it is unreasonable to assume that the internal structure of the DFA is known: indeed, this is exactly what is being inferred. In this work we present an active learning algorithm that makes few assumptions about the structure of the target DFA, and in fact outperforms one of the best heuristic methods for grammatical inference, which implicitly assumes that the DFAs are balanced (i.e. produce a more or less equal number of positive and negative labelings).

The current most powerful passive approach to grammatical inference using DFAs as models are the evidence driven state merging (EDSM) methods (see Cicchello and Kremer, 2003, for an overview), a heuristic approach that iteratively compresses an initially large DFA down to a smaller one, while preserving perfect classification before and after each compression. In this paper we compare our algorithm's performance against an EDSM variant implemented by Lucas and Reynolds (2005). Evolutionary approaches to grammatical inference also exist, in which a stochastic search method seeks the most accurate DFA model through mutation and recombination of previous models: in this work we will also compare our own method, which employs evolutionary computation for search, against the evolutionary method proposed by Lucas and Reynolds (2005). However, like the other passive methods, both heuristic and evolutionary approaches so far assume that some external agent generates either a random or balanced training set² before inference begins.

In the active learning approach to regular language inference pioneered by Angluin (1987) (see also Berg et al., 2003, and Angluin, 2004), the algorithm iteratively requests membership queries for training data it has generated on its own. Despite this active approach to training data generation, these algorithms also require an external agent—an oracle—that can answer equivalence queries: the oracle indicates whether the current model is equivalent to the target DFA and, if it is not, returns

1. See the definition of states and state transitions in Section 2.1 below.

2. A training set containing an equal number of positive and negative samples.

new training data that belongs to the target language but does not belong to the language encoded by the candidate model. Once again, this assumes that the oracle knows something about the structure of the target DFA. The algorithm presented here assumes that an oracle can answer membership queries, but not equivalence queries. In practical applications, such an oracle is the target system itself: the target system will return a classification for a proposed item, but cannot indicate whether a proposed model is equivalent to itself or not. The target system can indicate the goodness of a model if a large amount of sample data is classified by both itself and the proposed model and the resulting classifications are compared, but for target systems in which classifications are costly, slow or dangerous, this is not feasible.

Other active learning approaches to language inference also exist, but they all assume completely passive reception of training data: Sempere and Garcia (1993) only require that samples be presented in lexicographic order, and the RPNI (Oncina and Garcíá, 1992, and Lang et al., 1992) and RPNI2 Dupont (1996) algorithms assume random training data is supplied by an external agent, with the stipulation that positive and negative sample data must be made available.

The method presented in this paper does not assume any passive reception of training data from an external agent: rather, the algorithm attempts to evolve sentences that, when passed to the target system, should indirectly extract information about previously hidden components of the target system. For example, sentences should be sent to a target system that, during labelling, cause transitions to states that have never or rarely been visited during previous labellings. This is particularly useful in cases when passively-generated training data will cause some states of the target DFA to be visited much more often than others. In system identification, such systems are said to have low observability; it is more difficult to observe some components of the system than others using input data generated without recourse to a partial model of the system. For this and other reasons, it is not surprising that active learning approaches outperform passive methods: active methods have more control over the collection of training data. However the point of this paper is to demonstrate one reason why active methods outperform passive methods: namely, that they perform well on both balanced and imbalanced DFAs. More specifically, it is shown that one of the leading passive methods, the EDSM method, does poorly because it only performs well on balanced DFAs using balanced training data.

Large and unbalanced DFAs are one kind of automata that have low observability: these DFAs contain a large number of states, but tend to produce one labelling much more often than the other labelling, for any given sentence. For example one particular language (Tomita language 1, see Tomita, 1982) only produces a positive classification for a given binary string 2.4% of the time. In such cases, generating random training data is not recommended, because few or no sentences that elucidate the pathways to accepting states will be collected. Also, generating balanced training data is also not recommended, for two reasons. First, there will be a surfeit of training data elucidating paths to accepting states, and most likely not enough training data to elucidate the many other paths to non-accepting states, leading to the generation of a model that may have high training data accuracy but low test set accuracy. Secondly, generating balanced training data requires many labellings by the target system until a sufficient number of the minority labellings are collected. For example in order to obtain training data with 100 positively labelled data and 100 negatively labelled data for Tomita language 1, at least $\lceil \frac{100}{0.024} \rceil = 4167$ labellings of randomly generated sentences would have to be performed. This is not desirable for the real-world inference of languages or classifiers for which it is costly, dangerous or slow to perform a target labelling: the two performance

metrics for grammatical inference are model accuracy, and a minimum of sentence labellings by the target system.

Here we show that our algorithm outperforms competing methods that assume randomly-generated training data. For the case of imbalanced DFAs, we attribute this performance improvement to the discovery of sufficient minority class training data to produce accurate models: randomly-generated training data contains too little minority class training data. We support this claim by showing that the proposed algorithm performs well over a range of DFAs with differing balances (percentage of positive labellings), but that the EDSM method implemented here only performs well on DFAs within a narrow range of balances.

The fact that our algorithm also outperforms competing algorithms on balanced DFAs suggests that those DFAs contain state transition pathways that are rarely traversed by randomly-generated training data, but are better traversed by our proposed algorithm. However as of yet we have no supporting evidence for this stronger claim.

In the next section we briefly describe grammatical inference, as well as describing our method for the inference of target DFAs using active training data generation. We also document an evolutionary and a heuristics-based method for performing grammatical inference using pre-selected training data. In Section 3 we compare results from our algorithm against these algorithms for both randomly-generated DFAs, and randomly-generated DFAs that have differing balances. In the final section we provide some discussion and concluding remarks.

2. Methods

In this section we introduce grammatical inference, and outline three methods for approaching the problem: evidence-driven state merging, evolutionary approaches, and the estimation-exploration algorithm.

2.1 Grammatical Inference

A deterministic finite automata, or DFA, is a type of finite state automata that can be represented using the five-tuple (n, Σ, T, s, F) where n is the number of states, Σ is the alphabet of the encoded language, T is a transition function, s is the start state, and F is a set of final, or accepting states. Then, given some sentence made up of a string of symbols taken from the alphabet Σ , and beginning at the start state s , the first symbol is extracted from the sentence, and based on that symbol the sentence transitions to a new state as indicated by T . A deterministic finite automata follows the transition dictated by the current sentence symbol, the current state and the state transition function T with a probability of 1; probabilistic finite automata (which have not yet been investigated using our method) include probability distributions that denote the probabilities of transitioning to new states given the current sentence symbol, the current state and the state transition function.

After a state transition the next symbol is then extracted from the sentence, and based on T the sentence transitions to a new state. This process is continued until all symbols in the sentence have been exhausted. If the last state visited is a member of F , then the sentence receives a positive classification (the sentence belongs to the language); otherwise, a negative classification is assigned (the sentence does not belong to the language).

The quality of a grammatical inference algorithm is viewed as one that can produce some T' and F' (together referred to as a candidate DFA) that matches the labels of a pool of sentences that have already been labelled by the target DFA (the training set accuracy). The candidate DFA

should then produce a high classification accuracy when supplied with a different set of unlabelled sentences (test set accuracy). More specifically, quality can be measured in five ways: generation of an accurate model using a small set of training data; probability of learning the target language using little training data; continued performance for target DFAs with increasingly more states; consistent performance across DFAs with differing balances; and generation of an accurate DFA in the face of training set noise.

2.2 Evidence-Driven State Merging Algorithm

A family of algorithms collectively known as evidence-driven state merging algorithms (EDSMs) (Trakhtenbrot and Barzdin, 1973; Lang et al., 1998; Cicchello and Kremer, 2003) have been proposed that can infer some target DFA in which the number of states is unknown. EDSM algorithms operate by first generating an augmented prefix tree acceptor (APTA) which by definition perfectly classifies all sentences in the training set. Subsequent steps then involve merging states such that the DFA still maintains perfect training set accuracy. It has been shown that state merging tends to increase the test set accuracy of the reduced DFA. However in the face of incomplete training data, it is possible that an incorrect merge may occur: a merge that does not affect training set accuracy but does decrease test set accuracy. All of the work on EDSM algorithms is concerned with how merges should be performed in order to minimize test set accuracy degradation.

The EDSM method employed in this paper is adopted from (Lucas and Reynolds, 2005), which in turn modifies the EDSM method proposed by Price (Lang et al., 1998). The algorithm works as follows. Each pair of states in the APTA (or partially folded APTA) are considered for merging. The score of each merge is calculated by overlapping the roots and subtrees of the selected state pair, and summing the number of overlapped states that are either both accepting or rejecting states. If an accepting and rejecting state are found to overlap, that merge is disqualified. The state pair with the highest score is then selected for merging. In the case of a tie score, the state pair that appears first in the sequence of upper triangular matrix raster scan order $[(0, 1), (0, 2), \dots, (1, 2), (1, 3), \dots]$ is selected. The merge is then performed, and the previous steps are repeated until no further merges can be performed (*i.e.* all candidate merges are disqualified).

2.3 Evolutionary Approaches to Grammatical Inference

Evolutionary approaches to grammatical inference have also been proposed (Brave, 1996; Luke et al., 1999; Lucas and Reynolds, 2005). Generally, an evolutionary algorithm comprises a population of candidate models of the target DFA that compete against each other, and the fitness of a particular model is given by the percentage of training data that it can correctly classify. The model with the highest fitness at the termination of the run is then evaluated against the unlabelled test data. In this paper we compare our own evolutionary algorithm against the evolutionary method proposed by Lucas and Reynolds (2005). This approach is described below.

2.3.1 EVOLVING DFAS WITH A FIXED NUMBER OF STATES

Lucas and Reynolds (2005) proposed an evolutionary approach to grammatical inference in which the number of states in candidate models is fixed at $5n/4$, where n is believed to be the number of states in the target DFA. On target DFAs with $n \leq 16$ and a range of training set densities, this methodology outperforms the EDSM method outlined above (see Section 3).

<p>1. Characterization of the target system</p> <ul style="list-style-type: none"> • Define a representation, variation operators and similarity metric for the space of systems • Define a representation and variation operators for the space of inputs (tests) • Define a representation and similarity metric for the space of outputs <p>2. Initialization</p> <ul style="list-style-type: none"> • Create an initial population of candidate models (random, blank, or seeded with prior information) • Create an initial population of candidate tests (random, or seeded with prior information) <p>3. Estimation Phase</p> <ul style="list-style-type: none"> • Evolve candidate models; encourage diversity • Fitness of a model is its ability to explain all input-output data in training set <p>4. Exploration Phase</p> <ul style="list-style-type: none"> • Evolve candidate tests (input sets) • Fitness of a test is the disagreement it causes among good candidate models • Carry out best test on target system, add input/output data to training set <p>5. Termination</p> <ul style="list-style-type: none"> • Iterate estimation-exploration (steps 3-4) until the population of models converges on a sufficiently accurate solution, or the target system exhibits some desired behavior. • If no model is found, the search space may be inappropriate, or the target system may be inconsistent • If no good test is found, then either: <ul style="list-style-type: none"> – all good candidate models are perfect; – the search method for finding good tests is failing; or – the target system may be partially unobservable <p>6. Validation</p> <ul style="list-style-type: none"> • Validate best model(s) using unseen inputs • If validation fails, add new data to training set and resume estimation phase
--

Table 1: Estimation-Exploration Algorithm Overview

In this method a transition function T' is encoded as a $\Sigma \times \frac{5n}{4}$ matrix, and each element in T' , t'_{ij} , lies in the range $[0, \frac{5n}{4} - 1]$. Each column of T' corresponds to a particular state: the first column is regarded as the start state. During parsing, state transition is computed as follows. Transition to the state indicated by t'_{ij} , where i is the current state, and the current symbol from the input sentence corresponds to the j th letter in alphabet Σ .

Lucas and Reynolds (2005) realized that it is not necessary to evolve F' in addition to T' , but rather that it can be constructed indirectly from T' and the training data. For each state in T' , compute the ratio of positive and negative training sentences that terminate at that state: if more positive sentences terminate there, then consider that state an accepting state; otherwise, consider it a rejecting state.

Rather than employing a generational genetic algorithm, this method employs a multi-start hill climber. A random instance of T' is selected, and then a mutation is introduced: if the mutation causes a decrease in training set accuracy, revert to the original T' . Otherwise, keep the mutation, and perform another mutation. If 10,000 mutations have been attempted with no improvement, store the current T' and create a new random T' . Continue this process until a T' perfectly labels the training data, or until a total of 1,000,000 mutations have been attempted. Return the T' with the highest training set accuracy.

2.4 The Estimation-Exploration Algorithm

Both the heuristic and evolutionary method described above assume passive inference: a set of labelled data is presented to the algorithm, and the algorithm produces a candidate model of the target DFA. We have developed an active learning methodology for inferring DFAs (as well as other nonlinear target systems) which we refer to as the estimation-exploration algorithm (EEA). The algorithm is composed of two phases, as are most active learning systems (refer to Baram et al., 2004, for an overview of active learning): the estimation phase uses i instances of training data obtained from the target system to construct a set of candidate models; the exploration phase generates a new sentence (an instance of training data) that causes maximal disagreement among the candidate models. This new sentence is then supplied to the target system, and the estimation phase then begins again with $i + 1$ training data points. The estimation phase in our algorithm corresponds to the learning algorithm \mathcal{A} as described by Baram et al. (2004), and the exploration phase corresponds to the querying function Q . The utility of an active learning system corresponds to how well \mathcal{A} and Q perform together, compared to a control method where \mathcal{A} operates alone on randomly-generated unlabelled data (refer to Baram et al. (2004) for an overview of active learning).

The estimation-exploration algorithm is essentially a *co – evolutionary* process comprising two populations. One population is of candidate models of the target system, where a model’s fitness is determined by its ability to correctly explain observed data from the target system. The other population is of candidate unlabelled sentences, each of whose fitness is determined by its ability to cause disagreement among model classifications (thereby elucidating model uncertainties), or by exploiting agreement among models to achieve some desired output (thereby capitalizing on model certainties). The query by committee algorithm (Seung et al., 1992) first proposed that a good test is one that causes maximal agreement among a set of different candidate learners: however, the method by which differing yet accurate learners and disagreement-causing tests are generated was not given. In the estimation-exploration algorithm, evolutionary algorithms are used both to synthesize accurate yet differing models, as well as useful tests. If successful, the two populations challenge each other and drive an ‘arms-race’ towards inference of the model or towards eliciting some desired output from it. In previous papers (Bongard and Lipson, 2004b,a) we outlined a methodology for applying the estimation-exploration algorithm to other kinds of nonlinear target systems. The general methodology is given in Table 2.3.1, and the specific application to grammatical inference is given below.

2.4.1 CHARACTERIZATION OF THE TARGET SYSTEM

Like Lucas and Reynolds (2005), we choose to represent the target DFA and candidate models as $2 \times n$ integer matrices. For target DFAs with alphabets containing more than two elements, a larger matrix or a different encoding would be required. For the case of the target DFA, n is known. For

the work presented here, however, we assume that the learning algorithm does not know the number of states in the target system, but has some idea as to the upper bound. For the random target DFAs presented in Section 3.1, we compare our results against those of Lucas and Reynolds (2005), in which the number of states in a candidate model was fixed to be $\frac{5n}{4}$: we set the maximum number of states in a candidate model to be $2n$ in order to make less assumptions about the size of the target DFA. The second difference between our method and that of Lucas and Reynolds (2005) is that we exert selection pressure favoring smaller DFAs, in the hope of discovering more general models, as will be explained in the subsection documenting the estimation phase below.

It is always assumed that the first state is the start state. In addition, the target DFA contains an additional binary vector of length n that indicates whether state i is an accepting or rejecting state (F). For each candidate DFA model T' we compute F' using the method described in Section 2.3.1, as originally proposed by Lucas and Reynolds (2005). Due to the difficulties of devising a similarity metric between the target DFA and a given candidate model, we have chosen to denote similarity between a model and target DFA as the test set accuracy of the candidate model. If it were possible to define a similarity metric between the target and a model DFA, it would be possible to quantitatively determine how well an inference algorithm was doing by periodically measuring the similarity of candidate models against a target DFA. This would serve as a validation phase before using the algorithm in a practical application, where it is assumed there is no measure of target-model similarity, except for a model's ability to consistently match the classifications produced by the target.

In the exploration phase, for the grammatical inference problem a training item is considered to be an unlabelled binary sentence s' . Each sentence is represented as a binary vector of length s_{\max} , where s_{\max} is the maximum sentence length to be found in the training or test set. An additional integer variable l is selected from $[0, s_{\max}]$ with a uniform distribution, and indicates how long the encoded sentence is. If $l < s_{\max}$, then the trailing digits $[l, l + 1, \dots, s_{\max}]$ are ignored during the labelling of the sentence.

2.4.2 INITIALIZATION

The algorithm begins inference by generating an unlabelled sentence at random, which is then labelled by the target DFA. The training set, consisting of a single labelled sentence, is then provided to the candidate models in the estimation phase.

During the first pass through the estimation phase, a random population of candidate models is generated. In order to generate a pool of competing candidate models, the population of models in the estimation phase is partitioned into two equally-sized, reproductively isolated sub-populations: no candidate model can place offspring into the other sub-population. When the estimation terminates, the two most fit candidate models from each sub-population are provided to the exploration phase. This partition has no additional computational costs, as the two populations of models take the same time to evaluate as a single population with twice as many models.

During subsequent passes through the estimation phase, the two best models from the previous pass are introduced into their respective sub-populations. The remaining slots are filled with randomly-generated candidate models.

At the beginning of each pass through the exploration phase, the population is seeded with random binary sentences.

2.4.3 ESTIMATION PHASE

It is important to maintain a diverse set of candidate models in the estimation phase, so that disagreement among models can be measured effectively at the exploration phase. Many diversity maintenance techniques exist, but here we take the simplest approach based on evolving in two, separate niches. Starting with an initial population of p candidate models (with $p/2$ models in each of the two sub-populations), the population is evaluated, fit models are selected, copied and mutated, and less fit models are deleted. No recombination operators are employed in the current implementation. The pass continues for a fixed number of generations (g).

The fitness of a candidate model is given by

$$f_{T'} = 1 - \frac{\sum_{j=1}^i |t_j - m_j|}{i}, \quad (1)$$

where t_j is the labelling provided for the j th sentence by the target DFA, and m_j is the labelling provided for the same sentence by the model DFA. Then a candidate model that obtains $f_{T'} = 1$ perfectly labels all of the i training sentences seen so far, and models with lower values of $f_{T'}$ have lower accuracies. Within each sub-population, genomes are then sorted in order of decreasing fitness, such that the model with the highest fitness is at the top of the list of models within the sub-population, and the least fit model is at the bottom of the list. If two or more models have the same fitness, then those models are sorted among themselves based on the number of internal states that were visited during processing of all i training sentences seen so far, such that the topmost model in the subset used the least number of states, and the bottommost model used the most.

Once all of the models in both sub-populations have been evaluated, a pair of candidate models from within the same sub-population are selected. Each genome has an equal probability of being selected. The lower model in the sorted list is overwritten by a copy of the model higher up in the list. This ensures that models with higher fitness produce more offspring than models with lower fitness, and smaller models produce more offspring than models of larger size and equal fitness: selection pressure favors more accurate and more compact models. If both models have the same fitness and the same size, then each model in the pair has an equal probability of replacing, or being replaced by the other model. Also, this selection method ensures that the model with the best fitness and least size in each sub-population is never overwritten.

When a model is copied, it undergoes mutation. Mutation involves the selection of a random value t'_{ij} , and the replacement of the value found there by a new integer chosen from $[0, 2n - 1]$ with a uniform distribution.

A total of $\frac{3p}{8}$ pairs are selected for replacement and mutation in each sub-population at the end of each generation. Note that a genome may be selected more than once during the same generation, and that it may produce more offspring, or be overwritten by a more fit or smaller model. Also note that a mutated offspring may be selected for copying and further mutation during the same generation.

2.4.4 EXPLORATION PHASE

The exploration phase maintains a population of the same size as that of the estimation phase (p), and evolves candidate sentences for the same number of generations (g). At the end of each generation, $\frac{3p}{4}$ pairs of sentences are selected, copied and mutated as described in the previous section: the sentence with higher fitness is copied over the sentence with lower fitness, and the copied sentence is then mutated.

The fitness for a given sentence is set to the amount of disagreement that that sentence causes when labelled by a pool of k candidate models:

$$f_{s'} = 1 - 2 \left| 0.5 - \frac{\sum_{j=1}^k c_j}{k} \right|, \quad (2)$$

where c_j is the classification of the candidate sentence by model j . Sentences that do not differentiate between the candidate models—all models produce the same classification—obtain $f_{s'} = 0$ (poorest quality); sentences that produce the maximum classification variance obtain $f_{s'} = 1$ (best quality). This fitness function relies on the fact that the most agreement is equivalent to half of the models returning a negative classification, and the other half returning a positive classification. For target DFAs that do not produce binary classifications, this function would have to be generalized.

When a sentence is evolved that induces high classification variance, and that sentence is classified by the target DFA, then the resulting classification will usually lend support to $k/2$ candidate models during the next pass through the estimation phase, and provide evidence against the remaining half. It is important to note that for the experiments reported here, $f_{s'}$ can only assume two values: 0 or 1, hence there is no gradient within the search space. This is the simplest implementation of the algorithm. We expect that increasing the number of sub-populations or using other diversity maintenance techniques such as deterministic crowding (Mahfoud, 1995) would improve these results by inducing a gradient in the search space. Future work is planned to assess the performance benefit of population diversity.

Mutation is executed slightly differently from the estimation phase: with an equal probability, either the sentence or the length parameter l are selected for mutation. If the sentence is selected, a random bit is chosen and flipped; if l is chosen, l is reset to a random value in $[0, s_{\max}]$. In this way the algorithm can modify both the content and length of a candidate string.

2.4.5 TERMINATION

A typical run would terminate when the population of models converges. In the experiments reported here, however, the number of iterations was set to use exactly the same number of sentence labellings as the benchmark algorithms we compare it to require. The algorithm iterates t times, where t is the number of sentences in the training set used by the competing algorithm. This results in the labelling of t sentences by the target DFA, t passes through the estimation phase, and $t - 1$ passes through the exploration phase (the first sentence proposed for labelling is a random sentence). After the t th pass through the estimation phase, the most fit model from the first sub-population is output for validation purposes.

2.4.6 VALIDATION

Validation involves computing the accuracy of the best candidate model on a previously unseen set of test sentences.

3. Results

The estimation-exploration algorithm was compared against two sets of target DFAs: DFAs generated randomly in accordance with the method described in (Lang et al., 1998) for generating DFAs with differing sizes; and DFAs generated using a more generalized method that creates DFAs of differing sizes and balances.

3.1 Random DFAs

Sets of target DFAs of increasing size were generated for comparison between the EDSM method described in Section 2.2 (indicated as ‘EDSM’ in Figure 2), the evolutionary method proposed by Lucas and Reynolds (2005) (indicated as ‘Lucas’ in Figure 2), the estimation-exploration algorithm with the exploration phase disabled (random sentences are proposed to the target DFA and indicated as ‘Passive EEA’ in ensuing figures), and the estimation-exploration algorithm (indicated as ‘Active EEA’ in ensuing figures). Although the passive variant of the EEA proposes sentences to the target DFA for labeling, it is considered passive because it does not actively construct training data; rather, it outputs random training data. This stresses the importance of actively seeking informative sentences for target labeling.

As prescribed by the generative method introduced by Lang et al. (1998), the target DFAs were generated by creating random digraphs with $5n/4$, where n is the desired number of active states: an active state is one that is visited by at least one test or training sentence during labelling. Graphs are continually generated until one is produced which has a depth of exactly $2\log_2 n - 2$, where the depth of a DFA is determined to be the maximum over all states of the length of the shortest string which leads to that state:

$$d = \max_{(x|\frac{5n}{4})} \min_{(y|\text{strings leading to state } x)} \text{length}(y). \quad (3)$$

Once a target DFA is generated, each state is labelled as either accepting or rejecting with equal probability.

The total number of binary strings available for labelling is given as

$$S_{\text{total}} = \sum_{i=0}^{\lfloor (2\log_2 n) + 3 \rfloor} 2^i, \quad (4)$$

in accordance with the Abbadingo method (Lang et al., 1998) for generating random DFAs, where $\lfloor (2\log_2 n) + 3 \rfloor$ is the maximum possible string length. This approach ensures that all binary strings from the null string to length $\lfloor (2\log_2 n) + 3 \rfloor$ can be found in either the training or test set.

Strings selected for membership in the training set for the passive methods outlined in Sections 2.2 and 2.3.1 were selected at random (with a uniform distribution) from among this set of possible strings. Given a desired training set density d , the number of strings chosen for the training set can then be computed as $t = \lfloor dS_{\text{total}} \rfloor$.

In order to fairly compare our active learning method against passive methods, we have elected to equalize the number of labellings performed by the target DFA (*i.e.* the number of training sentences that are labelled), and the number of labellings performed by candidate models during inference. Because EDSM methods do not maintain a population of candidate models but rather iteratively compress a single one, in order to compare our method against the EDSM method outlined here we equalize the amount of labelled data that both algorithms have access to.

In Lucas and Reynolds (2005), the total number of candidate model labellings m is equal to the number of training sentences times the number of mutations considered during hill climbing: $m = \lfloor dS_{\text{total}} \rfloor \times 10^6 = t \times 10^6$, where d is training set density and t is the number of training sentences. In the estimation-exploration algorithm a total of

$$pgk(t-1) + pg \sum_{i=1}^t i$$

labellings are performed, where p is the population size and g is the number of generations for both phases, and thus pg indicates the total number of either sentences or models that are evaluated during a single pass through either phase. k indicates the number of candidate models output by the estimation phase,³ so $pgk(t-1)$ indicates how many labellings are performed during the $t-1$ passes through the exploration phase. The second term indicates how many labellings are performed during the t passes through the estimation phase: during the first pass there is only one labelling per candidate model; during the second pass there are two labellings per model; and so on.

We can ensure that our method performs the same or fewer model labellings as Lucas' method by arbitrarily setting $p = g$, and solving for p as follows:

$$pgk(t-1) + pg \sum_{i=1}^t i = m \quad (5)$$

$$p^2(2(t-1) + \sum_{i=1}^t i) = t \times 10^6 \quad (6)$$

$$p \cdot g = \lfloor \sqrt{\frac{t \times 10^6}{(2(t-1) + \sum_{i=1}^t i)}} \rfloor \quad (7)$$

Note that $k = 2$ here because we partition the estimation phase populations into two sub-populations.

3.1.1 THE EFFECT OF COMPRESSION PRESSURE ON INFERENCE

One advantage of the EEA over the EDSM methods is that it allows for both compression and expansion of models: EDSM methods only allow for compression at each step, and do not allow re-expansion. The advantage of this is illustrated in Figure 1, which reports the application of the EEA to a randomly-generated DFA with $n = 8$ states. Two other variants were also applied to the same DFA. The passive variant disables the exploration phase of the algorithm, so that at each cycle through the algorithm, a random sentence is output to the target DFA for labelling. The third variant is identical to the active variant, except for two modifications. First, the fitness function that favors smaller DFAs is disabled: when two candidate models are selected and both achieve the same training set accuracy, the first model is copied over the second model with a probability of 0.5, regardless of whether the second model is smaller than the first. Second, the maximum number of states that any candidate model can encode in this variant was increased from $2n = 16$ to 80. At the end of each pass through the estimation phase for each variant, the test set accuracy of the best candidate model output by the first sub-population is computed. Only the first 100 iterations through each variant are shown.

As can be seen in Figure 1a, the active variant outputs a model consistent with all training and test data after the 93rd pass through the estimation phase. The other two variants never produce a perfectly consistent model. Figure 1b indicates that the size-insensitive variant tends to output increasingly large DFAs that obtain better training set accuracy (data not shown), but there is no marked improvement in test set accuracy. Thus the added fitness component that favors smaller DFAs does confer some performance benefit by indirectly selecting for DFAs that can generalize beyond the training set better. Second, it is noted that the size of the best candidate model increases and decreases over the inference process in the active variant of the EEA. It has been found that models that solve all training data so far are gradually replaced by equally-fit but smaller models,

3. In the work reported here two models are output: the best model from each of the two sub-populations.

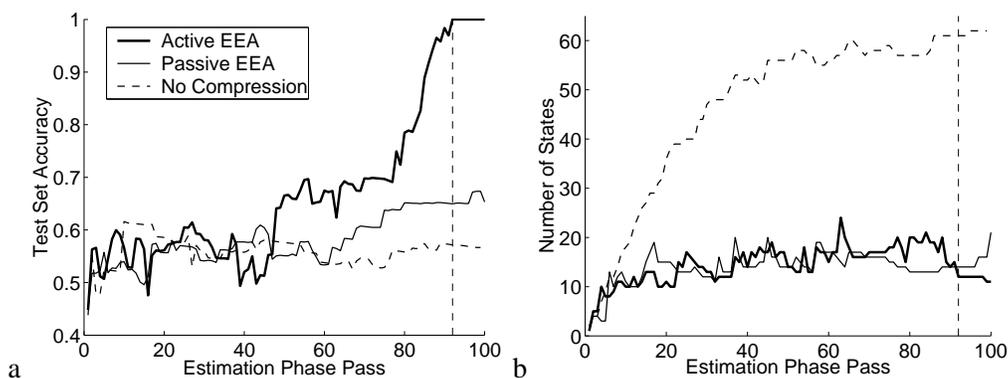


Figure 1: **Performance and Size of Candidate Models.** **a:** The test set accuracies of the best models output by each pass through the estimation phase. **b:** The total number of states used by each model when labelling the test sentences.

and new training data injected during the inference process causes older, smaller models to fail, only to be replaced with more accurate, larger models. This compression and expansion is a dynamic process that occurs as new training data is collected. Once a model consistent with all training data is obtained by the active EEA, subsequent passes through the estimation phase cause a compression of the candidate model (as long as the model is also consistent with the new training data): the model with $n = 10$ is reduced to $n = 9$ during the 99th pass through the estimation phase for the active EEA variant.

This example illustrates that evolutionary techniques are conducive to dynamic modelling in machine learning, at least in the specific case of grammatical inference, as they allow for dynamic restructuring of both the size of the model (the number of states) and its structure (the connections between states and whether a state is accepting or rejecting) as new training data is collected using active learning.

3.1.2 COMPARATIVE PERFORMANCE AMONG INFERENCE ALGORITHMS

Each of four algorithms—EDSM, Lucas’ method, and the active and passive variants of the EEA—were run against 3720 target DFAs: 1200 with $n = 4$, 1200 with $n = 8$, 1200 with $n = 16$, and 120 with $n = 32$ states. For the three smaller DFA classes, each algorithm was applied 100 times for each of 12 training set densities against a different DFA. For the large $n = 32$ DFA class, each algorithm was applied 10 times for each of 12 training set densities to a different DFA due to slower run times on these large DFAs.

The total number of training sentences available for inference for each DFA size and training set density is shown in Table 3.1.2. Table 3.1.2 reports the number of model labellings performed for each run of Lucas’ algorithm (left-hand figures) and the EEA (parenthesized figures).

Figure 2 reports the average performance of all four algorithms against the four DFAs and 12 training set densities. Performance is considered to be the test set accuracy of the best DFA output by each algorithm. The test set is comprised of all of the binary sentences that were not selected as training data. In the case of the estimation-exploration algorithm, which outputs two candidate

$d \setminus n$	4	8	16	32
0.01	2	10	40	163
0.02	5	20	81	327
0.03	7	30	122	491
0.04	10	40	163	655
0.05	12	51	204	819
0.06	15	61	245	982
0.07	17	71	286	1146
0.08	20	81	327	1310
0.09	22	91	368	1474
0.10	25	102	409	1638
0.15	38	153	614	2457
0.2	50	204	818	3276

Table 2: Total Numbers of Target Labellings

$d \setminus n$	4		8		16		32	
0.01	0.002	(0.002)	0.010	(0.010)	0.040	(0.040)	0.163	(0.162)
0.02	0.005	(0.005)	0.020	(0.019)	0.081	(0.080)	0.327	(0.321)
0.03	0.007	(0.007)	0.030	(0.029)	0.122	(0.121)	0.491	(0.483)
0.04	0.010	(0.010)	0.040	(0.040)	0.163	(0.162)	0.655	(0.653)
0.05	0.012	(0.012)	0.051	(0.050)	0.204	(0.200)	0.819	(0.810)
0.06	0.015	(0.015)	0.061	(0.060)	0.245	(0.242)	0.982	(0.981)
0.07	0.017	(0.017)	0.071	(0.070)	0.286	(0.279)	1.146	(1.108)
0.08	0.020	(0.019)	0.081	(0.080)	0.327	(0.321)	1.310	(1.243)
0.09	0.022	(0.022)	0.091	(0.090)	0.368	(0.365)	1.474	(1.412)
0.10	0.025	(0.024)	0.102	(0.100)	0.409	(0.403)	1.638	(1.555)
0.15	0.038	(0.037)	0.153	(0.151)	0.614	(0.595)	2.457	(2.371)
0.2	0.050	(0.049)	0.204	(0.200)	0.818	(0.808)	3.276	(3.095)

Table 3: Total Numbers of Model Labellings ($\times 10^9$)

models after the last pass through the estimation phase and then terminates, the best model is taken to be the model from the first sub-population. Figure 3 reports the percentage of runs of both the passive and active EEA variants that produced a model that correctly classifies all training and test data. The percentage of runs that produced such models for the various methods described in Lucas and Reynolds (2005) was not reported.

3.2 Unbalanced DFAs

As can be seen in Figure 2, the EDSM method only begins to compete with the active EEA for DFAs with $n = 32$ states. This seems to suggest that the EDSM methods scale better than the evolutionary method proposed here. However an alternate explanation of this observation is that the EDSM

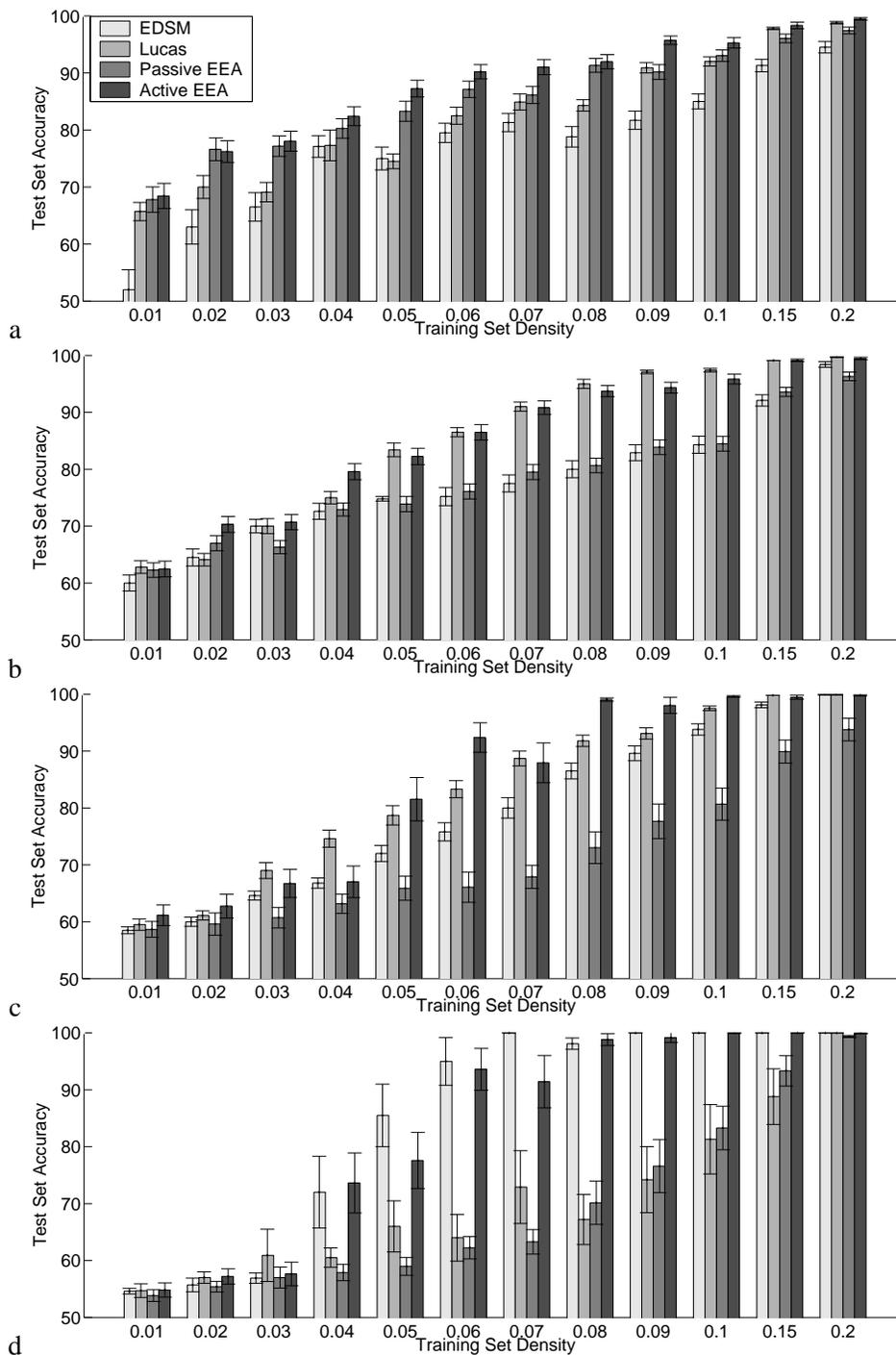


Figure 2: **Comparative Performance against Random DFAs.** The target DFAs are grouped according to size (**a**: $n = 4$, **b**: $n = 8$, **c**: $n = 16$, and **d**: $n = 32$) and training set density. Error bars indicate standard error computed over 100 runs for each algorithm for the first three DFA sizes (**a-c**), and over 10 runs for target DFAs with $n = 32$ (**d**).

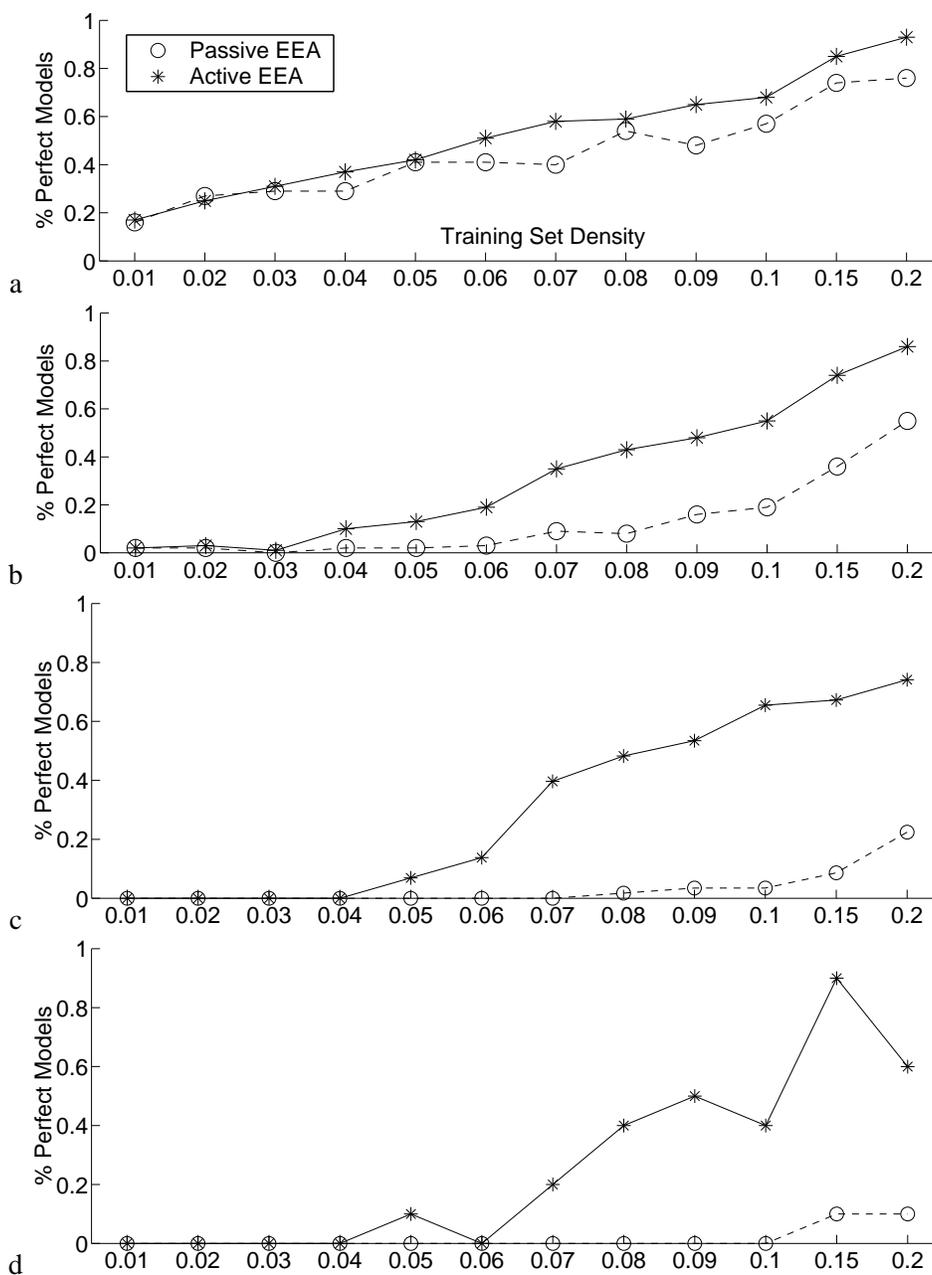


Figure 3: **Probability of Perfectly Consistent Model Discovery for the Random DFAs.** The target DFAs are grouped according to size (**a**: $n = 4$, **b**: $n = 8$, **c**: $n = 16$, and **d**: $n = 32$) and training set density. Data points indicate the percentage of runs that produced a model that achieves 100% test set accuracy.

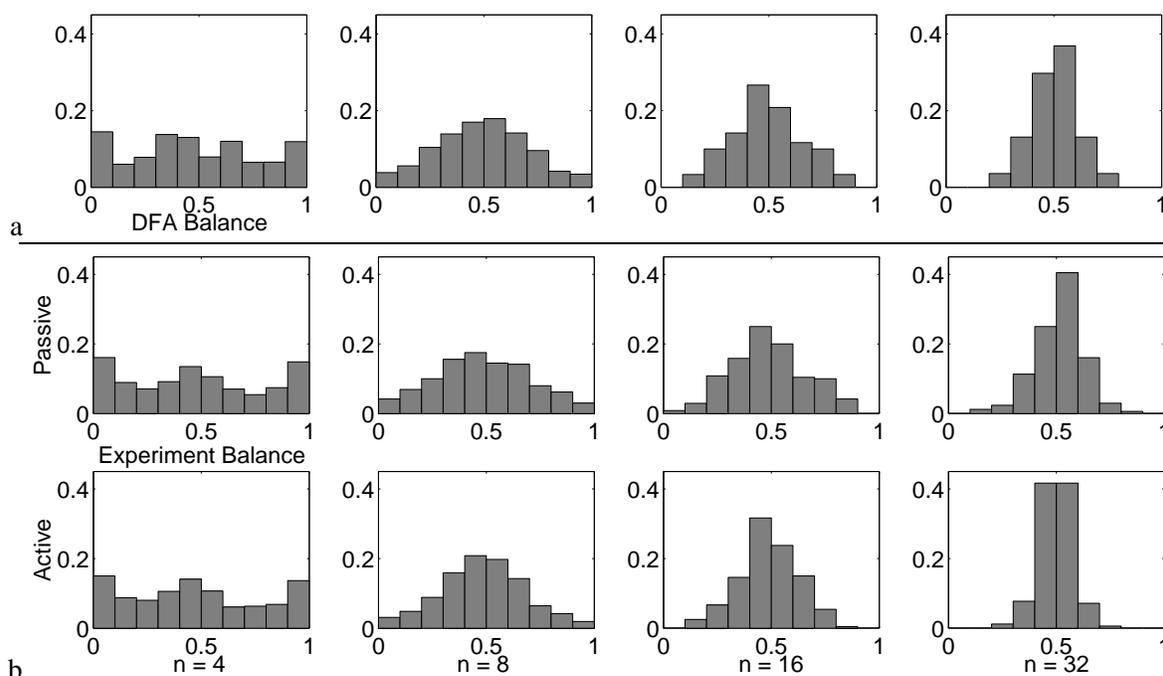


Figure 4: **a: Balance Distribution for the Random DFAs.** The 4 sets of 1200 random DFAs reported in Section 3.1 were grouped according to their size and balance. Balances were calculated using all random strings of length 0 to $\lfloor (2\log_2 n) + 3 \rfloor$. Each bar indicates the fraction of DFAs that fall within that particular range of balances. The upper row of panel *b* shows the distribution of balances for the same set of DFAs. Balances were calculated only using the strings output by the passive variant of the estimation-exploration algorithm applied to that DFA. The bottom row reports the balance distributions of the same DFAs using the active variant of the algorithm.

method works increasingly well on the large instances of the class of random DFAs produced by the generative method proposed by (Lang et al., 1998) and described in Section 3.1.

Figure 4a reports the balance distributions of the four DFA sizes produced by this generative method. As the figure indicates, DFAs with $n = 4$ states tend to exhibit a uniform distribution of balances, but as the DFAs increase in size the distribution clusters more closely around balanced DFAs that produce a more or less equal distribution of positive and negative labellings. For the largest class of DFAs ($n = 32$), the majority of DFAs have a balance within 0.4 and 0.6; the minority of DFAs produce less than 40% labellings of their minority classification.

This agrees with the original stated purpose of this generative approach, which was to produce random, balanced DFAs. However, this raises the possibility that methods developed to infer the DFAs produced by this method may not perform well on other kinds of DFAs, such as unbalanced DFAs. In order to test this, we generalized the generative method to produce DFAs of a desired size and balance. We then generated a number of DFAs of differing sizes and balances, and compared our algorithm against the EDSM algorithm described in Section 2.2. The new generalized method is as follows:

1. Select the desired number of states, n , and the desired balance b . The balance must be a real number in $[0, 1]$.
2. Create all random binary strings from length 0 to $\lfloor (2\log_2 n) + 3 \rfloor$.
3. Create a random digraph with $5n/4$ nodes.
4. While the depth of the graph is not $2\log_2 n - 2$, go to step 3.
5. For each state, label it as accepting with probability b ; otherwise, label it rejecting.
6. Pass each random string through the DFA, and compute the fraction of positive labellings. If the fraction of positive labellings is not in $[b - \epsilon, b + \epsilon]$, go to step 5. ϵ is taken to be some small tolerance; in the results reported below, ϵ is set to 0.01.

This method will produce DFAs with size centered around n states, and with a balance in $[b - \epsilon, b + \epsilon]$.

Using this method, a total of 15 DFAs were created: 5 with $n = 8$ states, 5 with $n = 16$ states, and 5 with $n = 32$ states. Within each size class, five DFAs were created with balances of 0.1, 0.2, 0.3, 0.4 and 0.5. For each DFA, the EDSM and the active variant of the EEA were applied 30 times using 12 different training set densities. The EDSM and EEA were instantiated using the same parameters described in the previous section.

Due to speed limitations, the EEA was not applied to the $n = 32$ DFAs using training set densities above 0.06. The mean test set accuracies of the EDSM and EEA methods are reported for the $n = 8$ DFAs in Figure 5, for the $n = 16$ DFAs in Figure 6, and for the $n = 32$ DFAs in Figure 7. The improvement factor for each DFA and corresponding training set density was calculated using $m_{\text{EEA}}/m_{\text{EDSM}}$, where m_{EEA} is the mean test set accuracy of the EEA for that DFA and that training set density, and m_{EDSM} is the mean accuracy for the same DFA and same training set density.

4. Discussion

Several trends can be noted in the mean performances of the algorithms reported in Figure 2. First, the evolutionary approach of Lucas and Reynolds (2005) tends to outperform the EDSM variant for smaller target DFAs ($n < 32$), but the EDSM far outperforms Lucas' algorithm for larger target DFAs ($n = 32$). Second, the active EEA variant outperforms all three other algorithms for the smallest size of target DFA ($n = 4$); is competitive with Lucas' algorithm for DFAs with $n = 8$ and $n = 16$; and is competitive with EDSM on the largest target DFAs ($n = 32$).

Third, the passive EEA variant performs poorly against the other three algorithms on all larger DFAs ($n > 4$). Because the passive variant performs the same or less model evaluations than Lucas' algorithm, and it randomly selects the same number of training sentences, we can conclude that our particular method of evolutionary search is inferior to that proposed by Lucas and Reynolds (2005). It seems plausible that replacing the evolutionary search that occurs within the EEA with a more powerful search technique—whether another evolutionary method, or a heuristic variant such as EDSM—may allow the proposed algorithm to outperform the passive forms of grammatical inference reviewed here.

The reason that the EDSM begins to compete with the EEA on the large $n = 32$ DFAs is made clear in Figure 7. The EDSM only performs well on DFAs centered at $b = 0.4$ (indicated by the

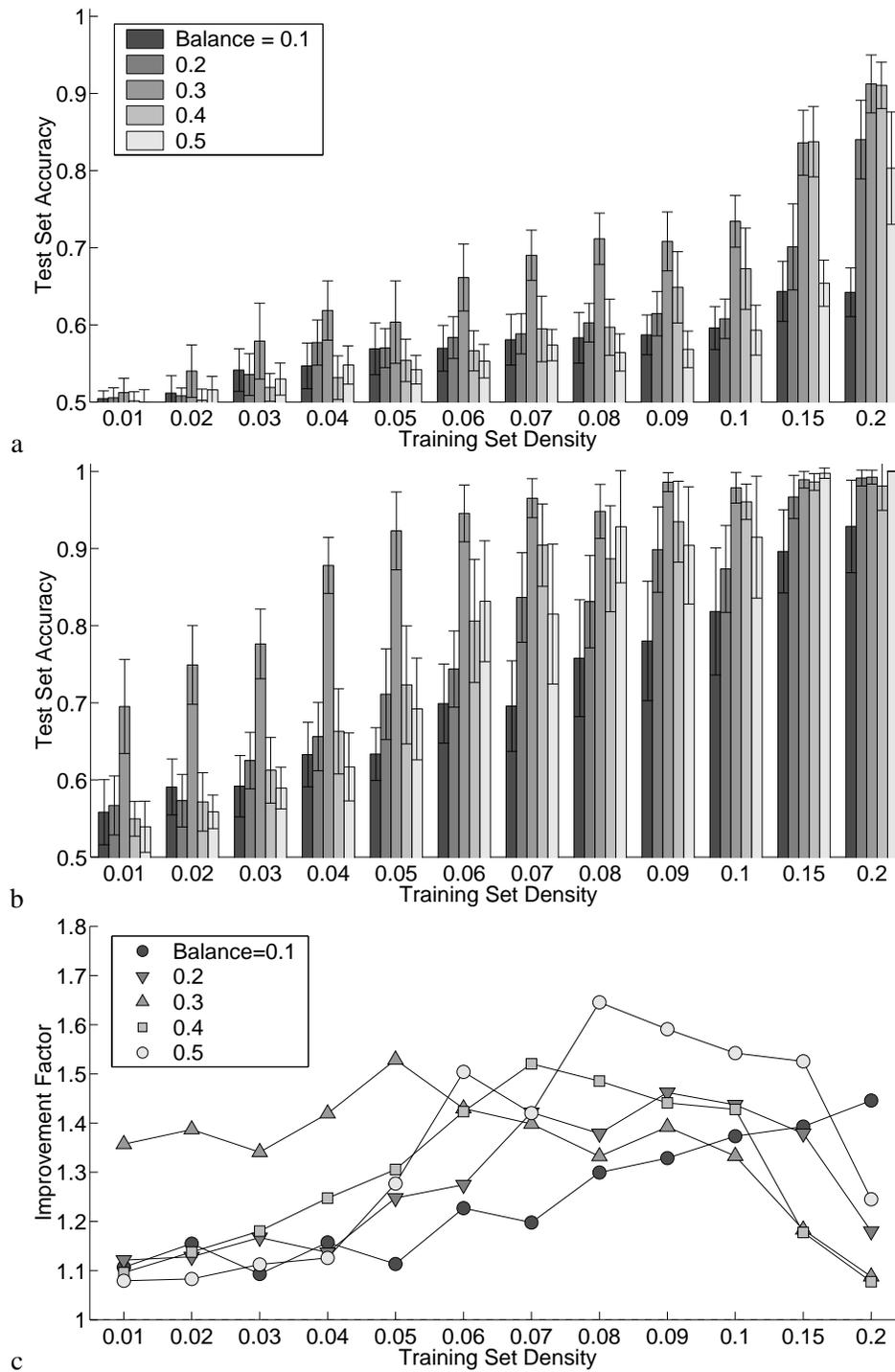


Figure 5: **Performance of the EEA against EDSM on DFAs with $n = 8$ states and differing balances.** **a:** Mean accuracies of the EDSM for differing balances and training set densities. **b:** Mean accuracies of the EEA for differing balances and training set densities. **c:** Improvement factors of the EEA over the EDSM for differing balances and training set densities.

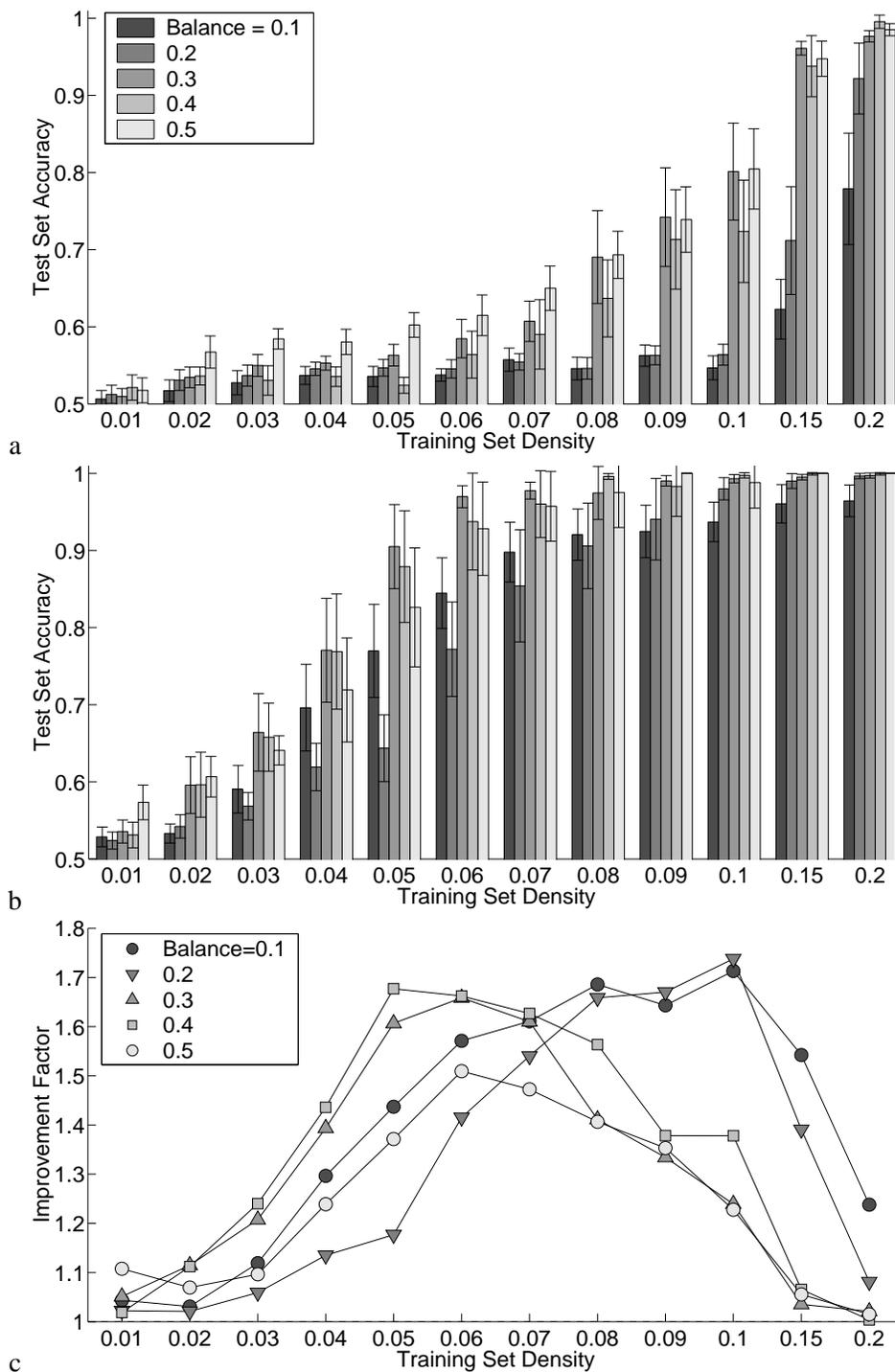


Figure 6: **Performance of the EEA against EDSM on DFAs with $n = 16$ states and differing balances.** **a:** Mean accuracies of the EDSM for differing balances and training set densities. **b:** Mean accuracies of the EEA for differing balances and training set densities. **c:** Improvement factors of the EEA over the EDSM for differing balances and training set densities.

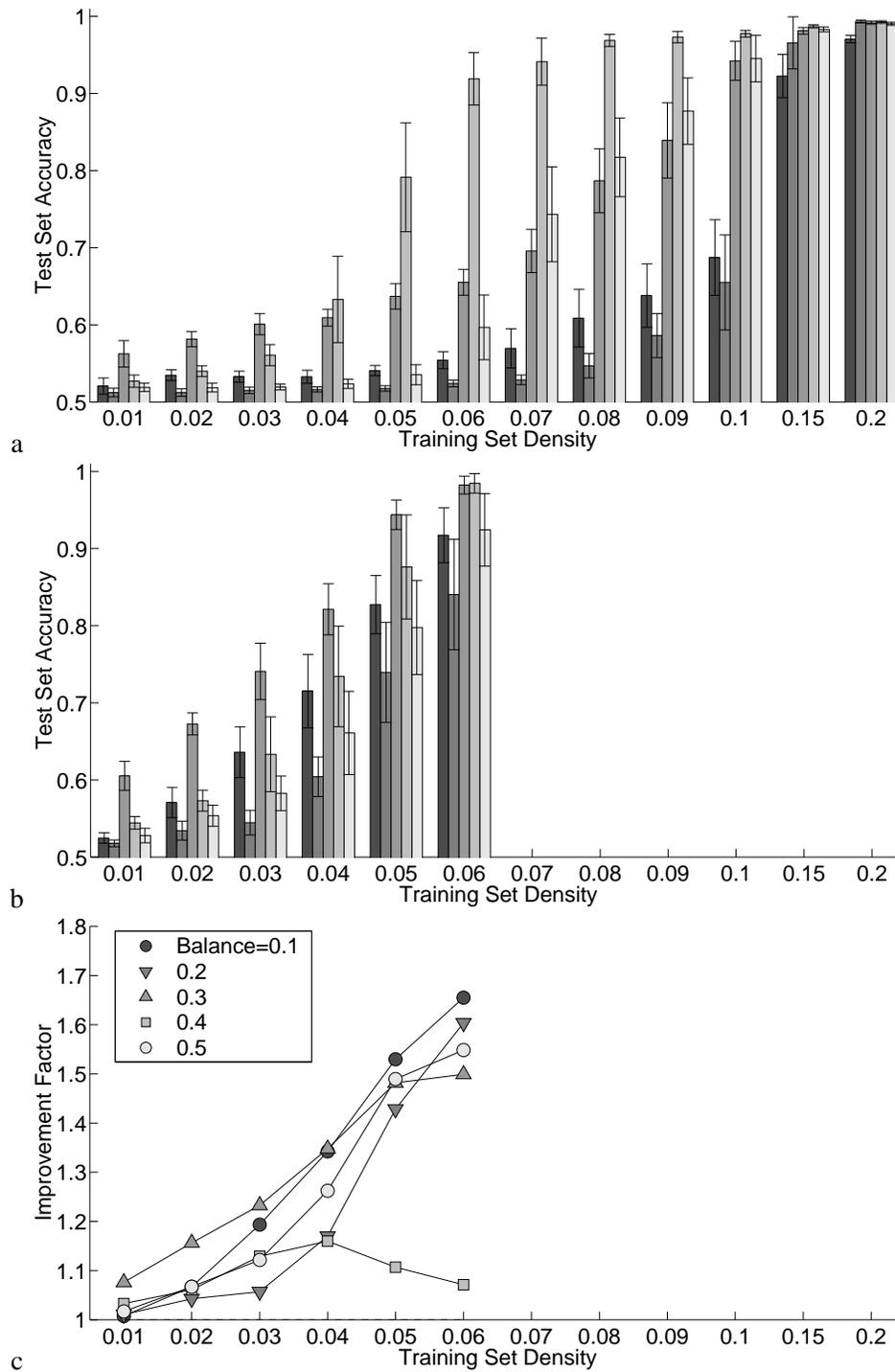


Figure 7: **Performance of the EEA against EDSM on DFAs with $n = 32$ states and differing balances.** **a:** Mean accuracies of the EDSM for differing balances and training set densities. **b:** Mean accuracies of the EEA for differing balances and training set densities. **c:** Improvement factors of the EEA over the EDSM for differing balances and training set densities.

balance= 0.4 bins in Figure 7a), and only slightly worse on DFAs with balances centered at $b = 0.5$ (indicated by the balance= 0.5 bins) for the intermediate training set densities. For low training set densities it does poorly on the DFAs of all balances, and the highest training set density it does well on DFAs of all balances. Finally, on the unbalanced DFAs centered around 0.1 and 0.2, it only begins to infer accurate models when supplied with training set densities of 0.15 and 0.2.

As Figure 4a illustrates, the generative method proposed by Lang et al. (1998) mostly produces random DFAs with balances in $[0.4, 0.6]$, which corresponds to the range of DFA balances for which the EDSM is well-suited. In contrast, as the EEA begins to perform successfully on the DFAs when allowed to generate sufficient training data, it begins to perform successfully on all of the DFAs, regardless of balance. As can be seen in Figure 7b, the standard deviations of the test set accuracies mostly overlap within each training set density class for the EEA, while the deviations of accuracies for the EDSM in Figure 7b do not overlap in many cases. This difference is highlighted by Figure 7c, which shows that for training set densities above 0.04, the EEA performs significantly better than the EDSM for all DFA balances except 0.4. From this it can be concluded that at least this EDSM variant performs well on just those DFAs with balances equal to those produced by the generative method proposed by Lang et al. (1998).

The balance specificity of the EDSM method is also clear in Figure 6a: EDSM does well for DFAs with balances of 0.3 and higher, but requires high training set densities to perform well on the DFAs with balances below 0.3. Alternatively, the EEA begins to perform better on all DFA balances as training set density increases (Figure 6b). Again this difference is highlighted by Figure 6c, which shows that for training set densities above 0.08, there is a significant performance increase of the EEA over the EDSM for the two imbalanced DFAs with $b = 0.1$ and $b = 0.2$.

In Figure 5, the balance specificity of the EDSM is less apparent. However, Figure 5c indicates that for DFAs with $b = 0.1$, the EEA achieves an increasing performance benefit over the EDSM for increasing training set densities (indicated by the increasing slope of the line with darkened circle markers). For instance, the EDSM only achieves a mean test set accuracy of 62% for the DFA with $n = 8$ and $b = 0.1$ using 0.2 training set density, while the EEA achieves a mean accuracy of 93% for the same DFA and the same amount of training data.

The reason why the active EEA infers imbalanced DFAs better than the EDSM is made clear by Figure 4b. For DFAs with $n = 32$ states, it can be seen that the active EEA generates training strings that achieve a more balanced labelling (lower righthand panel) than the passive EEA variant which outputs random strings for labelling (middle righthand panel). This is explained as follows. At the outset of inference using the active EEA, training strings are generated at random, because sufficiently accurate models do not yet exist. As inference proceeds, a few training strings are output to the target system that obtain a minority labelling. This allows for an increase in the accuracy of the set of candidate models in the estimation phase. Henceforth, training sets are evolved that cause disagreement among the models. This indicates that training strings with high fitness at least elicit a minority labelling from some of the candidate models, and since the models are now somewhat accurate, this increases the probability of obtaining a new minority labelling from the target system. As inference proceeds, minority labellings are extracted with increasing probability, allowing for the better inference of imbalanced DFAs.

This advantage of active training data generation, compared to passive training data collection, is also supported by the results reported in Figure 3. Clearly, the active EEA discovers models consistent with all training and test data more often than the passive EEA.

These results highlight the need to broaden the class of DFAs that are considered in grammatical inference. Furthermore, an inference algorithm should be judged not just on how well it infers a DFA of a given size and given training data, but how well an algorithm does on DFAs of differing sizes and balances. This requires rethinking how grammatical inference algorithms are compared: rather than simply providing them with training data already collected from a DFA, the algorithms should be free to request training data labelling, as is done in the active learning community (Baram et al., 2004).

4.1 Intelligent Testing

The results reported here support the claim that useful experiments (in the domain of grammatical inference, binary sentences) are those that elicit informative responses from the target system. What qualifies as ‘informative’, however, will vary across problem domains. Here we have stressed that one informative type of test are training data belonging to the minority class of an imbalanced DFA: automatically and actively generating such informative tests helps the algorithm to outperform other methods that rely on passively generated random training data.

It is important to note that there is no explicit reward in the exploration phase for such sentences. Rather, the ability to cause disagreement between alternative, approximate models means that $k/2$ of the models will yield the minority label for such a sample, and because these models are somewhat accurate (but not yet perfect), there is an increased probability that that sentence will actually elicit the minority label from the target DFA. This is a useful trait to have in an inference method, because what qualifies as an informative is domain dependent. For example in the domain of classification, training data that lie near the intersection of the decision boundaries of candidate classifiers would be more informative than data that lies on the same side of the decision boundaries.

There may be other kinds of informative sentences in grammatical inference that are unknowingly being favored by the active EEA variant. For example it seems plausible that for many DFAs, longer sentences are more informative than shorter sentences. It is clear that states closer⁴ to the start state in the transition function T will be visited more often than distant states, and longer sentences have a higher probability of reaching these distant states than shorter sentences do. Also, because longer sentences traverse more state transitions than shorter sentences and therefore have a better chance of uncovering differing transitions among candidate models, we predict that longer strings would tend to produce more disagreement among candidate models than shorter sentences can. So, we predict that the active EEA variant will propose, on average, longer training sentences than a passive algorithm will. Whether longer sentences truly are more informative than shorter ones, and whether longer sentences are actually favored by the active EEA variant has not yet been verified.

Cast in another light, informative tests tend to expose the unobservable parts of the target system, thus accelerating the inference process. In grammatical inference, unbalanced DFAs are less observable than balanced DFAs: there are either less states that produce the minority labelling than states that produce the majority labelling, or minority labelling states are more distant from the start state than majority labelling states. It follows then that passive grammatical inference approaches are inappropriate in these cases, for one of two reasons: either a balanced training set is assumed, in which case the minority class is grossly over-represented in the training data; or random training

4. Here, we assume that distance between states—more specifically, the distance between the start state and a given state—is viewed as the number of paths that exist between those states, and the mean length of those paths.

data is assumed, in which case the minority class is grossly under-represented. An active approach to grammatical inference, like the estimation-exploration algorithm, actively generates a training set that falls between these two extremes, and accelerates inference.

It also seems clear that most real-world systems will be unbalanced: they will not produce equal numbers of all label types for a randomly generated set of sample data. Also, acquiring a balanced training set will require a large number of target labellings in order to obtain enough of the minority labellings. As stated previously, the estimation-exploration algorithm is designed for inference using as few target trials as possible, because real world systems may be costly, dangerous or slow to query.

Furthermore, our method may be useful for indicating what kinds of training data is most useful for the inference of particular kinds of languages, by simply observing what kinds of sentences are generated by our method. However, this line of investigation has not yet been pursued.

4.2 Time Complexity

The running time of the estimation-exploration algorithm is proportional to the total number of labellings (trials) performed by the target DFA (t):

$$T(t) = gp \sum_{i=1}^t i + gtp^2 \quad (8)$$

$$= O(t^2), \quad (9)$$

where p and g are the population size and number of generations used by the genetic algorithm during each pass through either phases, respectively. The first term accounts for the running time of the estimation phase, which evolves models against all labellings seen so far. The second term accounts for the running time of the exploration phase, which evolves candidate trials and evaluates each one against each individual in the population of models, to estimate overall disagreement.

However, in the implementation of the EEA described here, a trial is not evaluated against all of the candidate models; a trial is only evaluated against the two best models from each of the two model sub-populations. This reduces overall running time but does not affect the time complexity of the algorithm:

$$T(t) = gp \sum_{i=1}^t i + 2gtp \quad (10)$$

$$= O(t^2). \quad (11)$$

Therefore, the algorithm running time increases polynomially with the total number of training strings presented to the target DFA.

However, the completion of the algorithm does not guarantee the output of a model that can correctly classify all training and test strings. Due to the complexity of the learning algorithm and its stochastic nature, we have not yet characterized the time complexity required to guarantee the output of such a consistent model. However, Figure 3 provides empirical evidence that for the random DFAs generated using the method proposed by Lang et al. (1998), a model DFA that correctly classifies all binary strings with lengths of 0 to $\lfloor (2\log_2 n) + 3 \rfloor$ can be found using the proposed algorithm. More specifically, for the active EEA, a model consistent with all training and test strings was found in at least 1 of the 100 runs for all training set densities for the $n = 4$ target

DFAs (Figure 3a); for all training set densities of 0.04 or higher for the $n = 8$ target DFAs (Figure 3b); for all training set densities of 0.05 or higher for the $n = 16$ target DFAs (Figure 3c); and for all training set densities of 0.07 or higher for the $n = 32$ target DFAs (Figure 3d). This translates to the ability of the algorithm to find perfectly consistent models when it is allowed to propose at least 2 binary strings for labelling by a $n = 4$ target DFA; at least 40 binary strings for labelling by a $n = 8$ target DFA; at least 204 binary strings for labelling by a $n = 16$ target DFA; and at least 1146 binary strings for labelling by a $n = 32$ target DFA (data taken from Table 3.1.2). This indicates that the ability of the active EEA to produce a model consistent with all training and test data scales polynomially with the amount of training data. However, as reported in Section 3.2, the balance of a DFA also has an effect on inference ability for both EDSM and EEA methods. Formulating time complexities for both methods as a function of both DFA size and balance as well as allowed number of target labellings requires further investigation.

5. Summary and Conclusions

Here we have introduced a co-evolutionary approach to grammatical inference that differs from both passive and active learning methods in that training data is not assumed to be provided by an external agent (either all at once or iteratively), but is generated internally by the algorithm itself: one component of the algorithm evolves a pool of candidate models, and the second component evolves a new training sentence that causes maximal disagreement among them. The sample that causes the most disagreement is then sent to the target language for labelling. We refer to this method as the estimation-exploration algorithm, or EEA. We have previously used this method to infer other kinds of tightly coupled, nonlinear target systems (see Bongard and Lipson, 2005, for an overview).

It has been shown here that the EEA outperforms another evolutionary approach to random DFA inference. Furthermore, the EEA outperforms the more powerful of the heuristic approaches (EDSM) on small random DFAs, and is competitive with EDSM on larger random DFAs.

The reason why the EDSM methods seem to perform better as the target DFAs increase in size was investigated. It was found that the EDSM method investigated here does not improve in ability as DFAs increase in size, but rather performs well on DFAs with specific balances (percentages of positive labellings), and that the generative method introduced by Lang et al. (1998) produces large DFAs with just these balances. It was shown that the EEA performs better on DFAs with differing balances by actively extracting minority labellings from the target DFA.

In order to better gauge the inference ability of grammatical inference methods, we introduce a more general method for generating DFAs with specific sizes and balances. In future, methods should be shown to work well not only on large DFAs with limited training data, but also consistently on DFAs of the same size but differing balances.

Our algorithm also allows for continual expansion and compression of candidate models over time, in response to new training data: expansion allows for the accommodation of new training data, and compression usually leads to greater test set accuracy. The current EDSM methods only allow for state compression, raising the possibility of inaccurate generalization. Another benefit of the EEA is that the internal search mechanism could be replaced with a more powerful search method: our algorithm functions independently of the search method used for inferring models and generating informative tests. It may be that replacing the current, basic evolutionary method with a more powerful stochastic search method (or even a deterministic one such as an EDSM variant) may

improve our method further. At the moment an upper bound is currently assumed on the number of states in a candidate model, but again, the current model search mechanism in EEA could be replaced by one that does not assume an upper bound, as was done in Luke et al. (1999).

In many approaches to grammatical inference, either a balanced training set is assumed, or random training data is generated. This can be wasteful when inference should be performed with as few labellings by the target language as possible, because either too little or too much of the minority training class is passively collected, or many labellings have to be performed in order to collect enough of the minority class training data for a balanced set. This is of practical importance because for many real-world languages or classifiers, collection of training data can be costly, dangerous or slow. In the EEA, only training sentences that cause maximal disagreement among the current set of candidate models are sent to the target DFA for labelling, and here we have shown that this process builds an informative training set: the set contains sufficient minority class training data to produce accurate models.

In future work we intend to apply our algorithm to probabilistic finite automata—automata that output probabilities as to which class(es) a sequence may belong, rather than absolute class assignments—as well as noisy sample data: evolutionary methods have previously proven to be well suited to dealing with probabilistic and noisy systems. One possible approach would be to evolve test sequences that cause the candidate models to disagree most in the class probabilities they predict the target system will output for that sequence. We also intend to apply our method to larger languages ($n \gg 32$) in order to provide evidence that our approach could be useful in real-world situations.

In closing we suggest that the grammatical inference community consider broadening the suite of target systems and target system generation methods in order to avoid biasing the development of new inference methods that only perform well on the target systems produced by a particular generative method.

Acknowledgments

This work was supported by the National Academies Keck Futures Grant for interdisciplinary research, number NAKFI/SIG07. This research was conducted using the resources of the Cornell Theory Center, which receives funding from Cornell University, New York State, federal agencies, foundations, and corporate partners.

References

- D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87, 1981.
- D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.
- D. Angluin. Queries revisited. *Theoretical Computer Science*, 313:175–194, 2004.
- Y. Baram, R. E. Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

- T. Berg, B. Jonsson, M. Leucker, and M. Saksena. Insights to Angluin’s learning. Technical Report 2003-039, Uppsala Universitet, 2003.
- F. Bergadano and D. Gunetti. *Inductive Logic Programming: From Machine Learning to Software Engineering*. MIT Press, Cambridge, MA, 1995.
- J. Bongard and H. Lipson. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. In *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, pages 169–176. IEEE Computer Society, 2004a.
- J. Bongard and H. Lipson. Automating genetic network inference with minimal physical experimentation using coevolution. In *Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO)*, pages 333–345. Springer, 2004b.
- J. Bongard and H. Lipson. Automating system identification using co-evolution. *IEEE Transactions on Evolutionary Computation*, 9(4):361–384, 2005.
- S. Brave. Evolving deterministic finite automata using cellular encoding. In *Genetic Programming 96: Proceedings of the First Annual Conference on Genetic Programming*, pages 39–44. MIT Press, 1996.
- O. Cicchello and S. C. Kremer. Inducing grammars from sparse data sets: A survey of algorithms and results. *Journal of Machine Learning Research*, 4:603–632, 2003.
- P. Dupont. Incremental regular inference. In L. Miclet and C. Higuera, editors, *Proceedings of the Third ICGI-96, Lecture Notes in Artificial Intelligence 1147*, pages 222–237, 1996.
- P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference? In *Proceedings of the Second International Colloquium on Grammatical Inference (ICGI’94)*, pages 25–37, 1994.
- K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In *Grammatical Inference (Lecture Notes in Artificial Intelligence 1433)*, pages 1–12. Springer-Verlag, 1992.
- K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In V. G. Honavar and G. Slutzki, editors, *Proceedings of the Fourth International Colloquium on Grammatical Inference: ICGI 1998*, pages 1–12, London, UK, 1998. Springer-Verlag.
- L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1999.
- S. M. Lucas and T. J. Reynolds. Learning deterministic finite automata with a smart state labelling evolutionary algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1063–1074, 2005.
- S. Luke, S. Hamahashi, and H. Kitano. “Genetic” programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1098–1105. Morgan Kaufmann, 13-17 1999.

- S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Urbana, IL, USA, 1995.
- J. Oncina and P. Garcíá. Inferring regular languages in polynomial update time. In *Pattern Recognition and Image Analysis*, pages 49–61. World Scientific, 1992.
- T. Pao and J. Carr. A solution of the syntactic induction-inference problem for regular languages. *Computer Languages*, 3:53–64, 1978.
- R. G. Parekh and V. G. Honavar. Efficient learning of regular languages using teacher supplied positive examples and learner generated queries. In *Proceedings of the Fifth UNB Conference on AI*, pages 195–203, 1993.
- R. G. Parekh and V. G. Honavar. An incremental interactive approach for regular grammar inference. In *Proceedings of the Third ICGI-96 (Lecture Notes in Artificial Intelligence 1147)*, pages 238–250. Springer Verlag, 1996.
- L. Pitt. Inductive inference, DFAs and computational complexity. In *Proceedings of the International Workshop on Analogical and Inductive Inference (Lecture Notes in Artificial Intelligence 397)*, pages 18–44. Springer Verlag, 1989.
- S. Porat and J. Feldman. Learning automata from ordered examples. *Machine Learning*, 7:109–138, 1991.
- J. M. Sempere and P. Garcia. A new regular language learning algorithm from lexicographically ordered complete samples. In *IEEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives*, pages 6/1–6/7, 1993.
- H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, pages 387–294, San Mateo, CA, 1992. Morgan Kaufman.
- M. Tomita. Dynamic construction of finite automata from examples using hill climbing. In V. G. Honavar and G. Slutzki, editors, *Proceedings of the Fourth Annual Cognitive Science Conference*, pages 105–108, 1982.
- B. A. Trakhtenbrot and Y. M. Barzdin. *Finite Automata Behavior and Synthesis*. North-Holland Publishing Company, Amsterdam, 1973.

Assessing Approximate Inference for Binary Gaussian Process Classification

Malte Kuss

Carl Edward Rasmussen

Max Planck Institute for Biological Cybernetics

Spemannstraße 38

72076 Tübingen, Germany

KUSS@TUEBINGEN.MPG.DE

CARL@TUEBINGEN.MPG.DE

Editor: Ralf Herbrich

Abstract

Gaussian process priors can be used to define flexible, probabilistic classification models. Unfortunately exact Bayesian inference is analytically intractable and various approximation techniques have been proposed. In this work we review and compare Laplace's method and Expectation Propagation for approximate Bayesian inference in the binary Gaussian process classification model. We present a comprehensive comparison of the approximations, their predictive performance and marginal likelihood estimates to results obtained by MCMC sampling. We explain theoretically and corroborate empirically the advantages of Expectation Propagation compared to Laplace's method.

Keywords: Gaussian process priors, probabilistic classification, Laplace's approximation, expectation propagation, marginal likelihood, evidence, MCMC

1. Introduction

In recent years models based on Gaussian process (GP) priors have attracted much attention in the machine learning community. Whereas inference in the GP regression model with Gaussian noise can be done analytically, probabilistic classification using GPs is analytically intractable, see Rasmussen and Williams (2006) for a general overview. Several approaches to approximate Bayesian inference have been suggested, including Laplace's method, Expectation Propagation (EP), variational approximations and Markov chain Monte Carlo (MCMC) sampling, some of these in conjunction with generalisation bounds, online learning schemes and sparse approximations (e.g. Neal, 1998; Williams and Barber, 1998; Gibbs and MacKay, 2000; Opper and Winther, 2000; Csató and Opper, 2002; Seeger, 2002; Lawrence et al., 2003).

Despite the abundance of recent work on probabilistic GP classifiers, most experimental studies provide only anecdotal evidence, and no clear picture has yet emerged, as to when and why which algorithm should be preferred. Thus, from a practitioners point of view it is unclear what the method of choice is for probabilistic GP classification. In this work, we set out to understand and compare two of the most wide-spread approximations: Laplace's method and Expectation Propagation (EP). We also compare to a sophisticated, but computationally demanding MCMC scheme, which becomes exact in the limit of long running times. We do not address issues of sparsification but stick to comparing the two types of approximation.

We examine two aspects of the approximation schemes: Firstly the accuracy of approximations to the marginal likelihood which is of central importance for model selection and model comparison.

In any practical application of GPs in classification (usually multiple) parameters of the covariance function (hyper-parameters) have to be handled. Bayesian model selection provides a consistent framework for setting such parameters. Therefore, it is essential to evaluate the accuracy of the marginal likelihood approximations as a function of the hyper-parameters, in order to assess the practical usefulness of the approach. The related question of whether the marginal likelihood correlates well with the generalisation performance cannot be answered in general but depends on the appropriateness of the model for a given data set. However, we do assess this empirically for two data sets.

Secondly, we need to assess the quality of the approximate probabilistic predictions. In the past, the probabilistic nature of the GP predictions has not received much attention, the focus being mostly on classification error *rates*. This unfortunate state of affairs is caused primarily by typical benchmarking problems being considered outside of a realistic context. The ability of a classifier to produce class probabilities or confidences, have obvious relevance in most areas of application, e.g. medical diagnosis and ROC analysis. We evaluate the predictive distributions of the approximate methods, and compare to the MCMC gold standard.

2. The Gaussian Process Model for Binary Classification

In this section we describe the Gaussian process model for binary classification (GPC). Let $y \in \{-1, 1\}$ denote the class label corresponding to an input \mathbf{x} . The GPC model is discriminative in the sense that it models $p(y|\mathbf{x})$ which for fixed \mathbf{x} is a Bernoulli distribution. The probability of success $p(y=1|\mathbf{x})$ is related to an unconstrained latent function $f(\mathbf{x})$ which is mapped to the unit interval by a sigmoidal transformation, e.g. the *logit* or the *probit*. Both mappings are relatively similar around zero but show different tail behaviour. We will not examine the difference in this study. For reasons of analytic convenience (for the EP algorithm) we exclusively use the probit model $p(y=1|\mathbf{x}) = \Phi(f(\mathbf{x}))$, where Φ denotes the cumulative density function of the standard normal distribution.

In the GPC model Bayesian inference is performed about the latent function f in the light of observed data $\mathcal{D} = \{(y_i, \mathbf{x}_i) | i=1, \dots, m\}$. Let $f_i = f(\mathbf{x}_i)$ and $\mathbf{f} = [f_1, \dots, f_m]^\top$ be shorthand for the values of the latent function and $\mathbf{y} = [y_1, \dots, y_m]^\top$ and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_m]^\top$ collect the class labels and inputs respectively.

Given the latent function, the class labels are independent Bernoulli variables, so the joint likelihood factorises:

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^m p(y_i|f_i) \tag{1}$$

and depends on f only through its value at the corresponding observed inputs. For the probit model the individual likelihood terms become $p(y_i|f_i) = \Phi(y_i f_i)$, due to the symmetry of Φ .

As prior over functions f we use a zero-mean Gaussian process (GP) prior (O’Hagan, 1978). A GP is a stochastic process where each input \mathbf{x} has an associated random variable $f(\mathbf{x})$. The joint distribution of function values corresponding to any set of inputs \mathbf{X} is multivariate Gaussian $p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$. The covariance matrix is defined element-wise, $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\theta})$ where k is a positive definite covariance function parameterised by $\boldsymbol{\theta}$. Note that by choosing a covariance function we introduce *hyper-parameters* $\boldsymbol{\theta}$ to the prior. The zero-mean GP prior encodes that *a priori* $p(y=1|\mathbf{x}) = 1/2$ and certain further beliefs about the characteristics of the latent function.

For details on covariance functions and their implications on the prior over functions see for example Abrahamsen (1997) or Rasmussen and Williams (2006, ch. 4).

Using Bayes' rule the posterior distribution over the latent function values \mathbf{f} for given hyper-parameters $\boldsymbol{\theta}$ becomes:

$$p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathcal{D}|\boldsymbol{\theta})} = \frac{\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})}{p(\mathcal{D}|\boldsymbol{\theta})} \prod_{i=1}^m \Phi(y_i f_i) \quad (2)$$

which is non-Gaussian. Properties of the posterior will be described in Section 5.

The main purpose of classification models is to predict the class label y_* for test inputs \mathbf{x}_* . The distribution of the latent function value can be computed by marginalisation:

$$p(f_*|\mathcal{D}, \boldsymbol{\theta}, \mathbf{x}_*) = \int p(f_*|\mathbf{f}, \mathbf{X}, \boldsymbol{\theta}, \mathbf{x}_*)p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta})d\mathbf{f}, \quad (3)$$

and by computing the expectation:

$$p(y_*|\mathcal{D}, \boldsymbol{\theta}, \mathbf{x}_*) = \int p(y_*|f_*)p(f_*|\mathcal{D}, \boldsymbol{\theta}, \mathbf{x}_*)df_* \quad (4)$$

the predictive distribution is obtained, which is again a Bernoulli distribution. The first term in the right hand side of equation (3) is Gaussian and obtained by conditioning the joint Gaussian prior distribution.

Unfortunately, neither the posterior eq. (2) $p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta})$, the predictive distribution eq. (4) $p(y_* = 1|\mathcal{D}, \boldsymbol{\theta}, \mathbf{x}_*)$ nor the marginal likelihood eq. (7) $p(\mathcal{D}|\boldsymbol{\theta})$ can be computed analytically, so approximations are needed. For the GPC model approximations are either based on a Gaussian approximation $q(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})$ to the posterior $p(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta})$ or involve Markov chain Monte Carlo (MCMC) sampling.

A key insight is that a Gaussian approximation to the posterior implies a GP approximation to the posterior process, which gives rise to an approximate predictive distribution for test cases. Introducing the approximate Gaussian posterior into eq. (3) gives the approximate posterior $q(f_*|\mathcal{D}, \boldsymbol{\theta}, \mathbf{x}_*) = \mathcal{N}(f_*|\mu_*, \sigma_*^2)$, with mean and variance:

$$\mu_* = \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{m} \quad (5a)$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K}^{-1} - \mathbf{K}^{-1} \mathbf{A} \mathbf{K}^{-1}) \mathbf{k}_*, \quad (5b)$$

where $\mathbf{k}_* = [k(\mathbf{x}_1, \mathbf{x}_*), \dots, k(\mathbf{x}_m, \mathbf{x}_*)]^\top$ is a vector of prior covariances between \mathbf{x}_* and the training inputs \mathbf{X} . For the probit likelihood the approximate predictive probability (4) of \mathbf{x}_* belonging to class 1 can be computed analytically:

$$q(y_* = 1|\mathcal{D}, \boldsymbol{\theta}, \mathbf{x}_*) = \int \Phi(f_*) \mathcal{N}(f_*|\mu_*, \sigma_*^2)df_* = \Phi\left(\frac{\mu_*}{\sqrt{1 + \sigma_*^2}}\right). \quad (6)$$

The parameters \mathbf{m} and \mathbf{A} of the posterior approximation can be found using Laplace's method (Section 3) or by Expectation Propagation (Section 4).

We have introduced the hyper-parameters $\boldsymbol{\theta}$ which we considered to be fixed. Typically very little information about these parameters is available *a priori*. In principle inference should be done jointly over f and $\boldsymbol{\theta}$ which can only be approximated using Markov chain Monte Carlo sampling.

However, a model selection approach can be implemented by selecting θ maximising the marginal likelihood (evidence):

$$p(\mathcal{D}|\theta) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}, \theta) d\mathbf{f} \quad (7)$$

which can be understood as a measure of the agreement between the model and observed data (Kass and Raftery, 1995; MacKay, 1999). This approach is called maximum likelihood II (ML-II) type hyper-parameter estimation and motivates the need for computing the marginal likelihood. Laplace’s method as well as Expectation Propagation provide an approximation to the marginal likelihood (7) and so approximate ML-II hyper-parameter estimation can be implemented in both approximation schemes.

3. Laplace’s Method

Williams and Barber (1998) describe Laplace’s method to find a Gaussian $\mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})$ approximation to the posterior over latent function values (2) for fixed θ (although they use the *logit* likelihood). Let $\ln \mathcal{L}(\mathbf{f}) = \ln p(\mathbf{y}|\mathbf{f})$ denote the log likelihood and:

$$\ln Q(\mathbf{f}|\mathcal{D}, \theta) = \ln \mathcal{L}(\mathbf{f}) - \frac{1}{2} \ln |\mathbf{K}| - \frac{1}{2} \mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f} - \frac{m}{2} \ln(2\pi) \quad (8)$$

the unnormalised log posterior. Laplace’s approximation is found by a second order Taylor expansion:

$$\ln Q(\mathbf{f}|\mathcal{D}, \theta) \simeq \ln Q(\mathbf{m}) - \frac{1}{2} (\mathbf{m} - \mathbf{f})^\top \mathbf{A}^{-1} (\mathbf{m} - \mathbf{f}) \quad (9)$$

around the mode of the (log) posterior:

$$\mathbf{m} = \operatorname{argmax}_{\mathbf{f} \in \mathbb{R}^m} \ln Q(\mathbf{f}|\mathcal{D}, \theta). \quad (10)$$

Since both the likelihood and the prior are log-concave the posterior is also log-concave and unimodal. Let:

$$\nabla_{\mathbf{f}} \ln Q = \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{f}) - \mathbf{K}^{-1} \mathbf{f} \quad (11a)$$

$$\nabla \nabla_{\mathbf{f}} \ln Q = \nabla \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{f}) - \mathbf{K}^{-1} \quad (11b)$$

denote the gradient and the Hessian. The mode is conveniently found using Newton’s method, iterating:

$$\mathbf{f} \leftarrow \mathbf{f} - (\nabla \nabla_{\mathbf{f}} \ln Q(\mathbf{f}))^{-1} \nabla_{\mathbf{f}} \ln Q(\mathbf{f}), \quad (12)$$

which usually converges rapidly to \mathbf{m} . The covariance matrix:

$$\mathbf{A} = -(\nabla \nabla_{\mathbf{f}} \ln Q(\mathbf{m}))^{-1} = (\mathbf{K}^{-1} + \mathbf{W})^{-1} \quad (13)$$

is approximated by the curvature at the mode, equal to the negative inverse Hessian, where $\mathbf{W} = -\nabla \nabla_{\mathbf{f}} \ln \mathcal{L}$.

This approximation also facilitates an approximation to the marginal likelihood:

$$p(\mathcal{D}|\theta) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}, \theta) d\mathbf{f} = \int \exp(\ln Q(\mathbf{f})) d\mathbf{f}. \quad (14)$$

Algorithm 1 Laplace’s approximation for GPC

Given: θ , \mathcal{D} , \mathbf{x}_*

Initialise \mathbf{f} (e.g. $\mathbf{f} \leftarrow \mathbf{0}$), compute \mathbf{K} from θ and \mathbf{X}

repeat

$\mathbf{f} \leftarrow \mathbf{f} - (\nabla \nabla_{\mathbf{f}} \ln Q(\mathbf{f}))^{-1} \nabla_{\mathbf{f}} \ln Q(\mathbf{f})$

until convergence of \mathbf{f}

m $\leftarrow \mathbf{f}$

A $\leftarrow (\mathbf{K}^{-1} - \nabla \nabla_{\mathbf{f}} \ln Q(\mathbf{m}))^{-1}$

Compute log marginal likelihood $\ln q(\mathcal{D}|\theta)$ by (15), and predictions $q(y_* = 1|\mathcal{D}, \theta, \mathbf{x}_*)$ using (6).

Substituting $\ln Q$ by its Taylor approximation (9) the Gaussian integral can be solved. The resulting approximate log marginal likelihood is:

$$\ln p(\mathcal{D}|\theta) \simeq \ln q(\mathcal{D}|\theta) = \ln Q(\mathbf{m}) + \frac{m}{2} \ln(2\pi) + \frac{1}{2} \ln |\mathbf{A}| \quad (15)$$

and the derivative of this quantity w.r.t. θ can be derived and used for optimisation (e.g. using conjugate gradient methods) in an ML-II type setting. See Algorithm 1 for an overview and Appendix A for details about our implementation.

4. Expectation Propagation

Minka (2001) proposed the iterative Expectation Propagation (EP) algorithm which can be applied to GPC. EP finds a Gaussian approximation $q(\mathbf{f}|\mathcal{D}, \theta) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})$ to the posterior $p(\mathbf{f}|\mathcal{D}, \theta)$ by moment matching of approximate marginal distributions. The starting point is an approximation mimicking the factorising structure:

$$p(\mathbf{f}|\mathcal{D}, \theta) = \frac{p(\mathbf{f}|X, \theta)}{p(\mathcal{D}|\theta)} \prod_{i=1}^m p(y_i|f_i) \simeq \frac{p(\mathbf{f}|X, \theta)}{q(\mathcal{D}|\theta)} \prod_{i=1}^m t(f_i, \mu_i, \sigma_i^2, Z_i) = q(\mathbf{f}|\mathcal{D}, \theta), \quad (16)$$

where throughout we use p to denote exact quantities and q approximations, and the terms:

$$t(f_i, \mu_i, \sigma_i^2, Z_i) = Z_i \mathcal{N}(f_i|\mu_i, \sigma_i^2) \quad (17)$$

are called *site functions*. Note that the site functions are approximating the likelihood (which normalizes over observations y_i), with a Gaussian in f_i , so we cannot expect the site functions to normalize, hence the explicit term Z_i is necessary. For notational convenience we hide the *site parameters* μ_i , σ_i^2 and Z_i and write $t(f_i)$ instead. From (17) the Gaussian approximation (16) has mean and covariance:

$$q(\mathbf{f}|\mathcal{D}, \theta) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A}), \text{ where } \mathbf{m} = \mathbf{A}\Sigma^{-1}\boldsymbol{\mu}, \text{ and } \mathbf{A} = (\mathbf{K}^{-1} + \Sigma^{-1})^{-1}, \quad (18)$$

where $\boldsymbol{\mu} = (\mu_1, \dots, \mu_m)^\top$ and $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_m^2)$ collect site function parameters. The EP algorithm iteratively visits each site function in turn, and adjusts the site parameters to match moments of an approximation to the posterior marginals. The k th moment of f_i under the posterior is:

$$\langle f_i^k \rangle = \frac{1}{p(\mathcal{D}|\theta)} \int f_i^k p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}, \theta) d\mathbf{f} = \frac{1}{p(\mathcal{D}|\theta)} \int f_i^k p(y_i|f_i) p_{\setminus i}(f_i) df_i \quad (19)$$

where:

$$p_{\setminus i}(f_i) = \int \prod_{j \neq i} p(y_j | f_j) p(\mathbf{f} | \mathbf{X}, \boldsymbol{\theta}) d\mathbf{f}^{\setminus i} \quad (20)$$

is called the *cavity distribution* and $\mathbf{f}^{\setminus i}$ denotes \mathbf{f} without f_i . The marginalisation required to compute the exact cavity distribution is intractable for the GPC model. The key step in the EP algorithm is to replace the intractable exact cavity distribution with a tractable approximation based on the site functions:

$$q_{\setminus i}(f_i) = \int \prod_{j \neq i} t(f_j) p(\mathbf{f} | \mathbf{X}, \boldsymbol{\theta}) d\mathbf{f}^{\setminus i}. \quad (21)$$

The approximate cavity function comes in the form of an unnormalised Gaussian $q_{\setminus i}(f_i) \propto \mathcal{N}(f_i | \mu_{\setminus i}, \sigma_{\setminus i}^2)$. Multiplying both sides by $t(f_i)$:

$$q_{\setminus i}(f_i) t(f_i) = \int \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K}) \prod_{j=1}^m t(f_j) d\mathbf{f}^{\setminus i} \propto \mathcal{N}(f_i | m_i, \mathbf{A}_{ii}), \quad (22)$$

and basic Gaussian identities give the parameters:

$$\sigma_{\setminus i}^2 = ((\mathbf{A}_{ii})^{-1} - \sigma_i^{-2})^{-1} \quad \text{and} \quad \mu_{\setminus i} = \sigma_{\setminus i}^2 \left(\frac{m_i}{\mathbf{A}_{ii}} - \frac{\mu_i}{\sigma_i^2} \right), \quad (23)$$

of the approximate cavity function.

The core idea of EP is to adjust the site parameters μ_i , σ_i and Z_i so that the approximate posterior marginal using the exact likelihood approximates as well as possible the posterior marginal based on the site function:

$$q_{\setminus i}(f_i) p(y_i | f_i) \simeq q_{\setminus i}(f_i) t(f_i, \mu_i, \sigma_i^2, Z_i) \quad (24)$$

by matching the zeroth, first and second moments. Recall that matching of moments minimizes Kullback-Leibler (KL) divergence.¹ For the probit likelihood $p(y_i | f_i) = \Phi(y_i f_i)$ the $k = 0, 1, 2$ moments of the left hand side can be computed analytically

$$m_0 = \Phi\left(\frac{y\mu_{\setminus i}}{\sqrt{1 + \sigma_{\setminus i}^2}}\right) = \Phi(z), \quad (25a)$$

$$m_1 = \mu_{\setminus i} + \frac{\sigma_{\setminus i}^2 \mathcal{N}(z | 0, 1)}{\Phi(z) y \sqrt{1 + \sigma_{\setminus i}^2}}, \quad (25b)$$

$$m_2 = 2\mu_{\setminus i} m_1 - \mu_{\setminus i}^2 + \sigma_{\setminus i}^2 - \frac{z \sigma_{\setminus i}^4 \mathcal{N}(z | 0, 1)}{\Phi(z) (1 + \sigma_{\setminus i}^2)}, \quad (25c)$$

where $z = y\mu_{\setminus i} / \sqrt{1 + \sigma_{\setminus i}^2}$. By equating these moments with those of the right hand side of (24) the update equations for the site parameters become

$$\sigma_i^2 = ((m_2 - m_1^2)^{-1} - \sigma_{\setminus i}^{-2})^{-1}, \quad (26a)$$

$$\mu_i = \sigma_i^2 \left(m_1 (\sigma_{\setminus i}^{-2} + \sigma_i^{-2}) - \frac{\mu_{\setminus i}}{\sigma_{\setminus i}^2} \right), \quad (26b)$$

$$Z_i = m_0 \sqrt{2\pi(\sigma_{\setminus i}^2 + \sigma_i^2)} \exp\left(\frac{(\mu_i - \mu_{\setminus i})^2}{2(\sigma_{\setminus i}^2 + \sigma_i^2)}\right). \quad (26c)$$

1. Although, the classical KL argument only applies to the first and second (and higher) moments for *normalized* distributions, it seems natural also to match zeroth moment.

Algorithm 2 EP for Gaussian process classification

Given: θ , \mathcal{D} , \mathbf{x}_*

Initialise: $\mathbf{A} \leftarrow \mathbf{K}$ and site parameters σ_i^2 and μ_i

repeat

for $i=1, \dots, m$ **do**

 Compute parameters (23) of cavity

 Compute moments (25)

 Update the site parameters using (26)

 Update \mathbf{m} and \mathbf{A} according to (18)

end for

until The site parameters converged

Compute log marginal likelihood $\ln q(\mathcal{D}|\theta)$ by (27), and predictions $q(y_* = 1|\mathcal{D}, \theta, \mathbf{x}_*)$ using (6).

In the application of EP, one may generally not have a guarantee that the new site variance in (26a) is non-negative; however, in the GPC model with probit likelihood, one can show that variance is always positive. Once we have new values for μ_i and σ_i^2 we have to update \mathbf{m} and \mathbf{A} according to (18), which in practise is done using rank-one updates, to save computation.

The EP algorithm iteratively updates the site parameters as shown in Algorithm 2. Although we cannot prove the convergence of EP, we conjecture that it always converges for GPC with probit likelihood, and have never encountered an exception.

Finally the approximate log marginal likelihood can be obtained from the normalization of (16), giving

$$\begin{aligned} \ln p(\mathcal{D}|\theta) &\simeq \ln q(\mathcal{D}|\theta) = \ln \int q(\mathbf{f}|\mathbf{X}, \theta) \prod_{i=1}^m t(f_i) d\mathbf{f} \\ &= \sum_{i=1}^n \ln Z_i - \frac{1}{2} \ln |\mathbf{K} + \Sigma| - \frac{1}{2} \boldsymbol{\mu}^\top (\mathbf{K} + \Sigma)^{-1} \boldsymbol{\mu} - \frac{m}{2} \ln(2\pi). \end{aligned} \quad (27)$$

Perhaps this is not the standard way to compute an approximation to the marginal likelihood used elsewhere, but it seems the most natural given the approximation. The derivatives of the log marginal likelihood can be computed in order to implement ML-II parameter estimation of θ . Algorithm 2 summarises the computations, more details on implementing EP for GPC can be found in Appendix B.

5. Structural Properties of the Posterior

In the previous sections we described the GPC model and two alternative approximation schemes for finding a Gaussian approximation to the posterior. This section provides more details on the properties of the posterior which is compared to the structure of the respective approximations.

Figure 1(a) provides a one-dimensional illustration. The prior $\mathcal{N}(f|0, 5^2)$ combined with the probit likelihood ($y = 1$) results in a skewed posterior. Intuitively, the likelihood cuts off the f values which have the opposite sign of y . The mode of the posterior remains relatively close to the origin, while the mass is placed over positive values in accordance with the observation. Laplace's approximation peaks at the posterior mode, but places far too much mass over negative values of f and too little over large positive values. The EP approximation attempts to match the first two

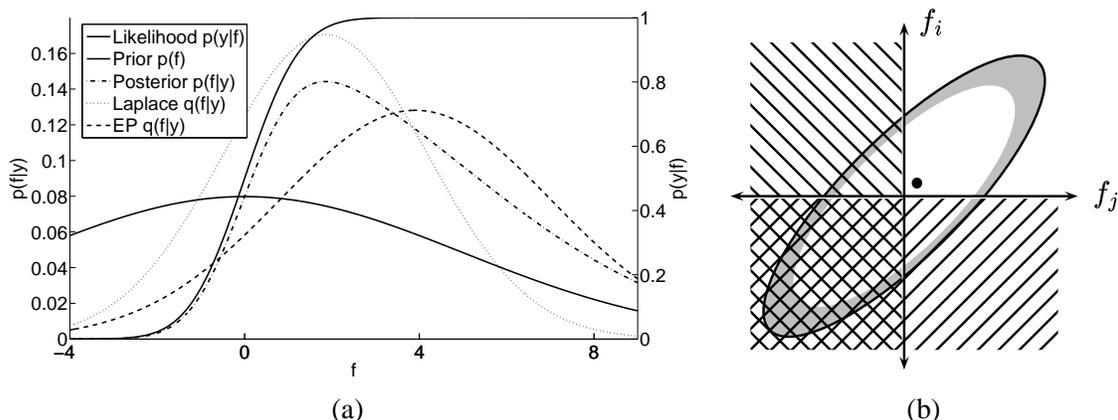


Figure 1: Panel (a) provides a one-dimensional illustration of approximations. The prior $\mathcal{N}(f|0, 5^2)$ combined with the probit likelihood ($y = 1$) results in a skewed posterior. The likelihood uses the right axis, all other curves use the left axis. In Panel (b) we caricature a high dimensional zero-mean Gaussian prior as an ellipse. The gray shadow indicates that for a high dimension Gaussian most of the mass lies in a thin shell. For large latent signals, the likelihood essentially cuts off regions which are incompatible with the training labels (hatched area), leaving the upper right orthant as the posterior. The dot represents the mode of the posterior, which is relatively unaffected by the truncation and remains close to the origin.

posterior moments, which results in a larger mean and a more accurate placement of probability mass compared to Laplace’s approximation.

Structural properties of the posterior in higher dimensions can best be understood by examining its construction. The prior is a correlated m -dimensional Gaussian $\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$ centred at the origin. Each likelihood term $p(y_i|f_i)$ softly truncates the half-space from the prior that is incompatible with the observed label, see Figure 1(b). The resulting posterior is unimodal and skewed, similar to a multivariate Gaussian truncated to the orthant containing \mathbf{y} . The mode of the posterior remains close to the origin, while the mass is placed in accordance with the observed class labels. Additionally, high dimensional Gaussian distributions exhibit the property that most probability mass is contained in a thin ellipsoidal shell—depending on the covariance structure—away from the mean (MacKay, 2003, ch. 29.2). Intuitively this occurs since in high dimensions the volume grows extremely rapidly with the radius. As an effect the mode becomes less representative (typical) for the prior distribution as the dimension increases. For the GPC posterior this property persists: the mode of the posterior distribution stays relatively close to the origin, still being unrepresentative for the posterior distribution, while the mean moves to the mass of the posterior making mean and mode differ significantly.

As described, we cannot generally assume the posterior to be close to Gaussian, as in the often studied limit of low-dimensional parametric models with large amounts of data. Therefore in GPC we must be aware of making a Gaussian approximation to a non-Gaussian posterior. Laplace’s approximation is centred around the mode of the posterior, which lies in the right orthant but too close

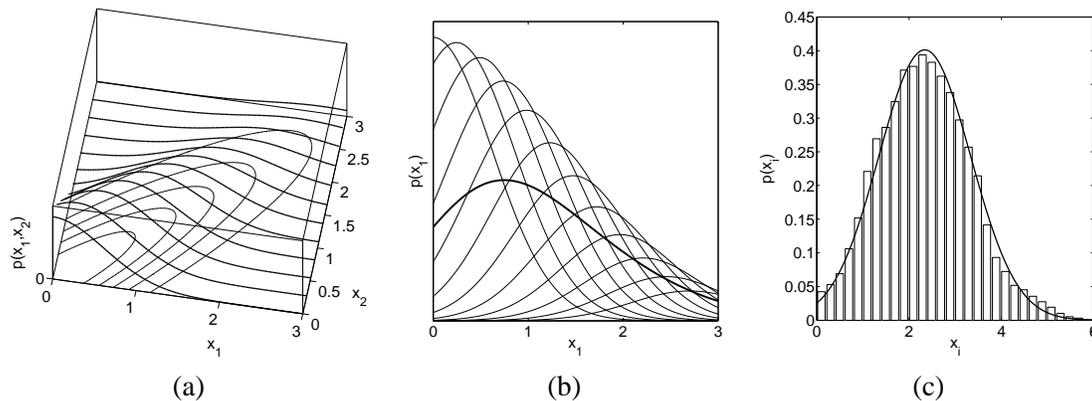


Figure 2: Panel (a) illustrates a bivariate normal distribution truncated to the positive quadrant. The lines describe slices through the probability density function for fixed x_2 -values. Panel (b) shows the marginal distribution of $p(x_1)$ (thick line) obtained by (numerical) integration over x_2 , which—intuitively speaking—corresponds to an averaging of the slices (thin lines) from Panel (a). Panel (c) shows a histogram of samples of a marginal distribution of an high-dimensional truncated Gaussian. The line describes a Gaussian with mean and variance estimated from the samples.

to the origin, such that the approximation will overlap with regions having practically zero posterior mass. As an effect the amplitude of the approximate latent posterior GP will be underestimated systematically, leading to overly cautious predictive distributions.

The EP approximation does not rely on a local expansion, but assumes that the marginal distributions of the posterior can be well approximated by Gaussians. As described above the posterior is similar to a high dimensional multivariate normal distribution truncated to one orthant. Although the posterior is skew and truncated, marginals of such a distribution can be relatively similar to a Gaussian.

As a low dimensional illustration the marginal distribution of a bivariate normal is shown in Figure 2(a-b). Depending on the covariance structure, the mode of the marginal distribution moves away from the origin and the distribution appear similar to a truncated univariate Gaussian.

In order to inspect the marginals of a truncated high-dimensional multivariate normal distribution we made an additional synthetic experiment. We constructed a 767 dimensional Gaussian $\mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{C})$ with a covariance matrix having one eigenvalue of 100 with eigenvector $\mathbf{1}$, and all other eigenvalues are 1. We then truncate this distribution such that all $\mathbf{x}_i \geq 0$. Note that the mode of the truncated Gaussian is still at zero, whereas the mean moved towards the remaining mass. Metropolis-Hastings sampling was used to generate samples from this truncated multivariate distribution. Figure 2(c) shows a normalised histogram of samples from a marginal distribution of one \mathbf{x}_i . The samples agree very well with a Gaussian approximation. Note that Laplace’s method would be completely inappropriate for approximating a truncated multivariate normal distribution.

In order to validate the above arguments we will use Markov chain Monte Carlo methods to generate samples from the posterior and also to estimate the marginal likelihood.

6. Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) may be too slow for many practical applications, but has the advantage that it becomes exact in the limit of long runs. Thus, MCMC can provide a *gold standard* by which to measure the two analytic methods of the previous sections. Computing the predictions via an MCMC estimate of (3) and (4) is relatively straight forward and covered in Section 6.1.

Good MCMC estimates of the marginal likelihood are, however, notoriously difficult to obtain, being equivalent to the free-energy estimation problem in physics (Gelman and Meng, 1998). In Section 6.2 we explain the use of Annealed Importance Sampling (AIS), which can be seen as a sophisticated elaboration of Thermodynamic Integration, for this task.

6.1 Hybrid MCMC Sampling

Hybrid Monte Carlo (HMC) sampling as proposed by Duane et al. (1987) is a computationally efficient sampling technique which exploits gradient information of the target distribution. Detailed accounts are given by Neal (1993, ch. 5.2) and Liu (2001, ch. 9). MacKay (2003, ch. 30) also provides pseudo-code; we do not repeat the details here.

HMC can be used to generate samples from the posterior $p(\mathbf{f}|\boldsymbol{\theta}, \mathcal{D})$, while only the unnormalised log posterior (8) and its derivatives are required. As described in the previous section, the exact posterior (2) takes the form of a (correlated) Gaussian (the GP prior), which is (softly) truncated by the constraints imposed by the training labels through the likelihood. To ease the sampling task by reducing correlations, we first do a linear transformation into new $\mathbf{g} = \mathbf{L}^{-1}\mathbf{f}$ variables, such that \mathbf{g} is *white* w.r.t. \mathbf{K} , where $\mathbf{K} = \mathbf{L}\mathbf{L}^\top$ is the Cholesky decomposition. Given samples from the posterior, we generate test-latents from the Gaussian $p(f_*|\mathbf{f}, \mathbf{X}, \boldsymbol{\theta}, \mathbf{x}_*)$ for use in a simple Monte Carlo estimate of (4).

6.2 Annealed Importance Sampling

The marginal likelihood (7) comes in the form of an m dimensional integral where m is the number of data points. A simple approach would be to use importance sampling with the EP or Laplace’s approximation of the posterior as proposal distribution. However, for the GPC model the resulting importance weights show enormous variances, making simple importance sampling useless for this task (MacKay, 2003, ch. 29).

Neal (2001) describes Annealed Importance Sampling (AIS), which we will use to estimate the marginal likelihood in the GPC model. Instead of solving the integral (7) directly, a sequence of easier quantities is computed. We define:

$$Z_t = \int p(\mathbf{y}|\mathbf{f})^{\tau(t)} p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) d\mathbf{f} \quad (28)$$

where $\tau(t)$ is an inverse temperature schedule such that $\tau(0) = 0$ and $\tau(T) = 1$. The trick is to rewrite the marginal likelihood $Z = p(\mathcal{D}|\boldsymbol{\theta})$ as a fraction and expand:

$$Z = \frac{Z_T}{Z_0} = \frac{Z_T}{Z_{T-1}} \frac{Z_{T-1}}{Z_{T-2}} \dots \frac{Z_1}{Z_0}, \quad (29)$$

Algorithm 3 Annealed Importance Sampling

Given: Temperature schedule τ
for $r = 1, \dots, R$ **do**
 Sample \mathbf{f}_0 from the prior $\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$
 for $t = 1, \dots, T$ **do**
 Sample \mathbf{f}_t from $q(\mathbf{f}|\mathcal{D}, \tau(t), \boldsymbol{\theta})$ by HMC
 Compute $\ln(Z_t/Z_{t-1})$ using (31)
 end for
 Compute Z_r using (32)
end for
 Return $\ln Z = \ln \left(\frac{1}{R} \sum_{r=1}^R Z_r \right)$

where $Z_0 = 1$ since the prior normalises. Each term in (29) is approximated using importance sampling using samples from $q(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}, \tau(t)) \propto p(\mathbf{y}|\mathbf{f})^{\tau(t)} p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$:

$$\frac{Z_t}{Z_{t-1}} = \int \frac{p(\mathbf{y}|\mathbf{f})^{\tau(t)} p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{f})^{\tau(t-1)} p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})} q(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}, \tau(t-1)) d\mathbf{f} \quad (30a)$$

$$\simeq \frac{1}{S} \sum_{i=1}^S p(\mathbf{y}|\mathbf{f}_i)^{\tau(t)-\tau(t-1)} \quad (30b)$$

where \mathbf{f}_i are samples from $q(\mathbf{f}|\mathcal{D}, \boldsymbol{\theta}, \tau(t))$, which we generate using HMC. Using a single sample $S = 1$ and a large number of temperatures, the log of each ratio is:

$$\ln(Z_t/Z_{t-1}) \simeq (\tau(t) - \tau(t-1)) \ln p(\mathbf{y}|\mathbf{f}_t) \quad (31)$$

where \mathbf{f}_t is the only sample at temperature $\tau(t)$. Combining (29) with (31) we obtain the desired:

$$\ln Z \simeq \sum_{t=1}^T \ln(Z_t/Z_{t-1}). \quad (32)$$

In all our experiments we use $\tau(t) = (t/T)^4$ for $t = 0, \dots, 8000$. Using this temperature schedule we found that the sampling spends most of its efforts at temperatures with high variance of (31) such that the variance of (32) is relatively small. Note that this was only examined on the data sets we use below and only for certain values of $\boldsymbol{\theta}$. So far, we have described Thermodynamic Integration, which gives an unbiased estimate in the limit of slow temperature changes. In AIS the bias caused by finite temperature schedules is removed by combining multiple estimates by their geometric mean (see Algorithm 3). In the experiments we combine the estimates of $R = 3$ runs of Thermodynamic Integration.

7. Experiments

In this section we compare and inspect approximations for GPC using various benchmark data sets. The primary focus is not to optimise the absolute performance of GPC models but to compare the relative accuracy of approximations and to validate the arguments given in Section 5.

In all the GPC experiments we use a covariance function of the form:

$$k(\mathbf{x}, \mathbf{x}', \boldsymbol{\theta}) = \sigma^2 \exp \left(-\frac{1}{2\ell^2} \|\mathbf{x} - \mathbf{x}'\|^2 \right), \quad (33)$$

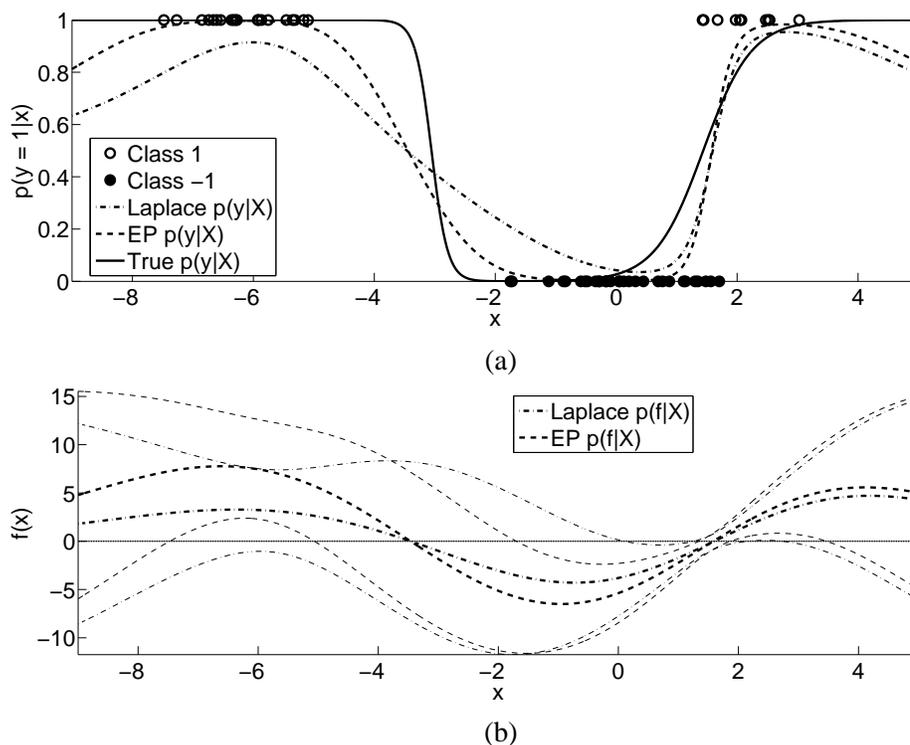


Figure 3: Synthetic classification problem: Panel (a) illustrates the classification task, the generating $p(y|x)$ and two approximations thereof obtained by Laplace’s method and EP. Panel (b) illustrates the approximate predictive distributions $p(f_*|\mathcal{D}, \theta, x_*) \simeq \mathcal{N}(f_*|\mu_*, \sigma_*^2)$ of latent function values showing the mean μ_* and the range of $\pm 2\sigma_*$.

such that $\theta = [\sigma, \ell]$. We refer to σ^2 as the signal variance and to ℓ as the characteristic length-scale. Note that for many classification tasks it may be reasonable to use an individual length scale parameter for every input dimension (ARD). Nevertheless, for the sake of presentability we use the above covariance function and we believe the conclusions to be independent of this choice.

Both analytic approximations have a computational complexity which is cubic $O(m^3)$ as common among non-sparse GP models due to inversions $m \times m$ matrices. In our implementations Laplace’s method and EP need similar running times, on the order of a few minutes for several hundred data-points. Making AIS work efficiently requires some fine-tuning and a single estimate of $p(\mathcal{D}|\theta)$ can take several hours for data sets of a few hundred examples, but this could conceivably be improved upon.

7.1 Synthetic Classification Problem

The first experiment is a synthetic classification problem with scalar inputs. The observations for class 1 were generated from two normal distributions with means -6 and 2 , each with a standard deviation of 0.8 . For class -1 the mean is 0 and the same standard deviation was used.

We computed Laplace’s and the EP approximation for the ML-II estimated value of θ that maximised Laplace’s approximation to the marginal likelihood (15). Note that this particular choice of θ should be in favour of Laplace’s method. Figure 3 shows the resulting classifiers and the underlying latent functions. In Figure 3(a) the approximations to $p(y|x)$ appear to be similar for positive x but we observe an appreciable discrepancy for negative values. Laplace’s approximation gives an unreasonably high predictive uncertainty, which is caused by a significant overlap of the approximate predictive distribution $p(f_*|\mathcal{D}, \theta, x_*) \simeq \mathcal{N}(f_*|\mu_*, \sigma_*^2)$ with zero as shown in Figure 3(b). However, note that both approximations agree on the sign of the predictive mean.

7.2 Ionosphere Data

The data consists of 351 examples in 34 dimensions. We standardised the inputs \mathbf{X} to zero mean and unit variance. The training set is a random subset of size $m = 200$ leaving the remaining 151 instances out as a test set.

We do an exhaustive investigation on a regular 21×21 grid of values for the log hyper-parameters. For each θ on the grid we compute the approximated log marginal likelihood by Laplace’s method (15), EP (27) and AIS. Additionally we compute the predictive performance on the test set. As performance measure we use the average information in bits of the predictions about the test targets in excess of that of random guessing. Let $p^* = p(y_* = 1|\mathbf{x}_*)$ be the model’s prediction, then we average:

$$I(p_i^*, y_i) = \frac{y_i+1}{2} \log_2(p_i^*) + \frac{1-y_i}{2} \log_2(1 - p_i^*) + H \tag{34}$$

over all test cases, where H is the entropy of the training set labels. Results are shown in Figure 4.

For all three approximation techniques we see an agreement between marginal likelihood estimates and test performance, which justifies the use of ML-II parameter estimation. But the shape of the contours and the values differ between the methods. The contours for Laplace’s method appear to be *slanted* compared to EP. The estimated marginal likelihood estimates of EP and AIS agree very well.² The EP predictions contain as much information about the test cases as the MCMC predictions and significantly more than for Laplace’s method.

Note that for small signal variances (roughly $\ln(\sigma^2) < 0$) Laplace’s method and EP give very similar results. A possible explanation is that for small signal variances the likelihood does not *truncate* the prior but only *down-weights* the tail that disagrees with the observation. As an effect the posterior will be less skewed and both approximations will lead to similar results.

7.3 USPS Digits

We define a binary sub-problem from the USPS digit data³ by considering 3’s vs. 5’s. We repeated the experiments described in the previous section for a slightly modified grid of θ . Comparing the results shown in Figure 5 leads to similar results as mentioned above. The EP and MCMC results agree very well, given that the marginal likelihood comes as a 767 dimensional integral.

We now take a closer look at the approximations $q(\mathbf{f}|\mathcal{D}, \theta) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})$ for a given value of θ . We have chosen the values $\ln(\sigma) = 3.35$ and $\ln(\ell) = 2.85$ which are between the ML-II estimates of EP and Laplace’s method. Comparing the respective means of the approximations in Figure 6(a) we

2. Note that the agreement between the two seems to be limited by the accuracy of the AIS runs, as judged by the regularity of the contour lines; the tolerance is less than one unit on a (natural) log scale.
 3. Because the training and test partitions in the original data differ significantly, we pooled cases and randomly divided them into new sets, with 767 cases for training and 773 for testing.

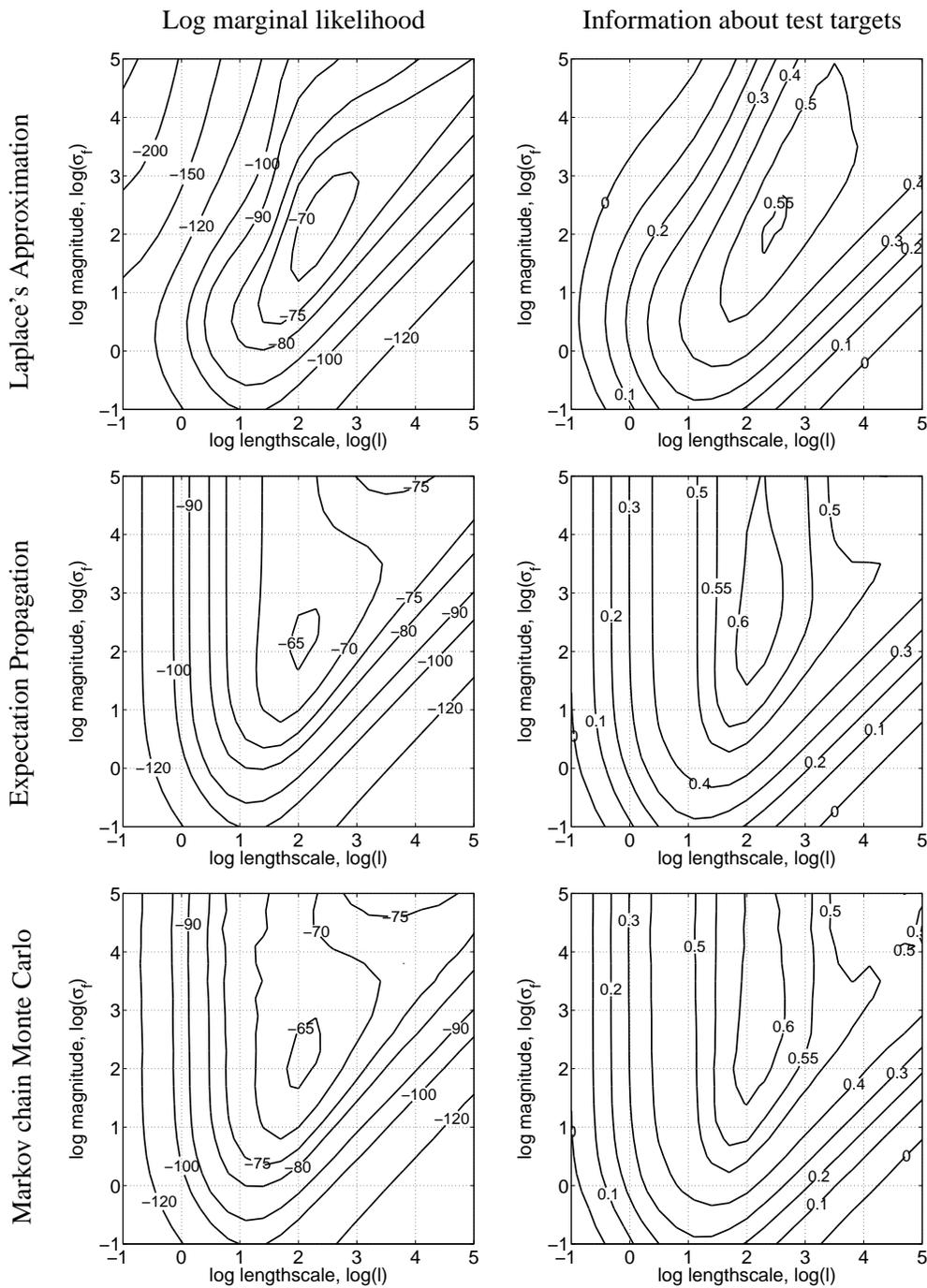


Figure 4: Comparison of marginal likelihood approximations and predictive performances for the Ionsphere data set. The first column shows the estimates of log marginal likelihood, while the second column shows the performance on the test set measured by the information about test targets in bits (34).

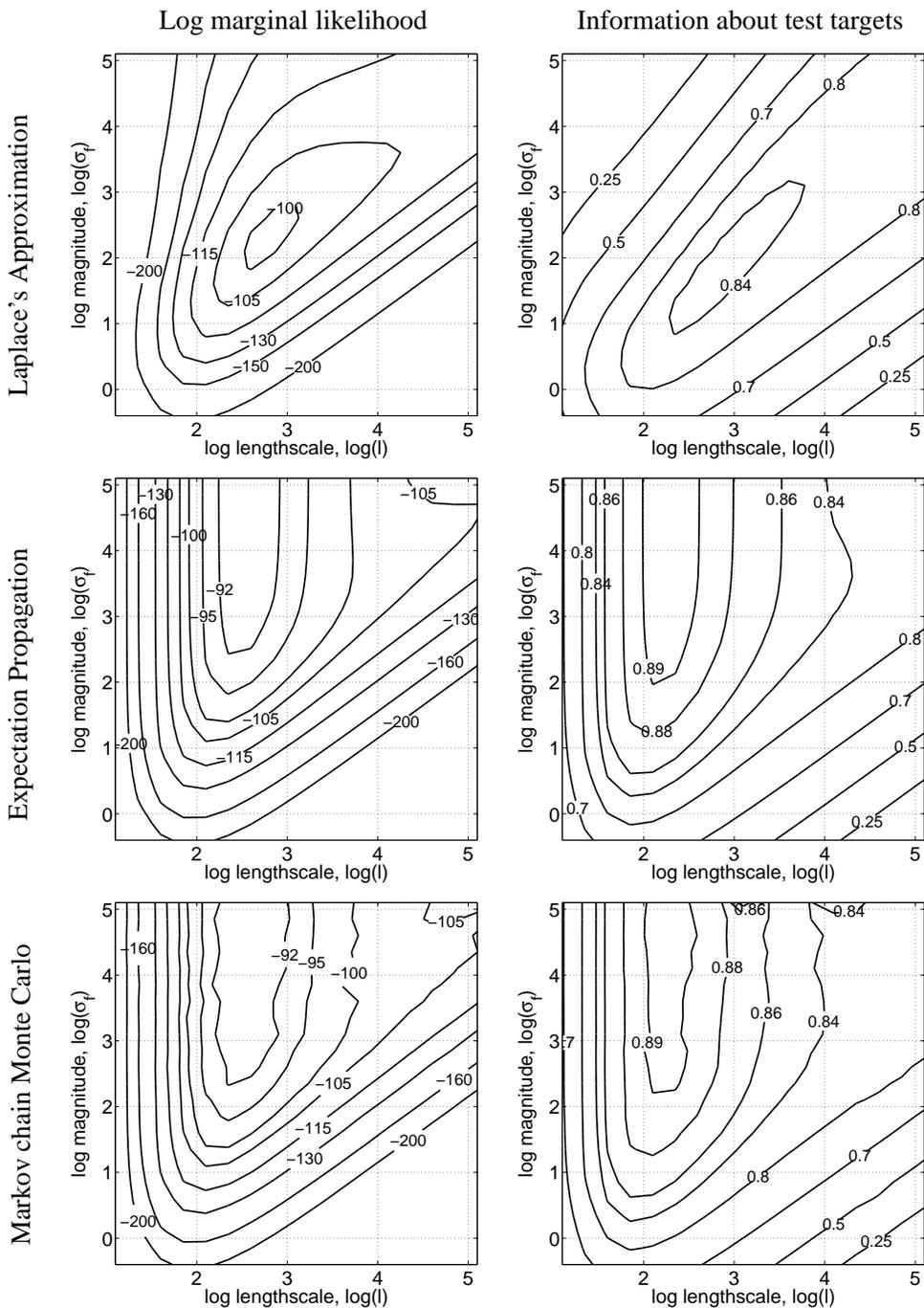


Figure 5: Comparison of marginal likelihood approximations and predictive performances of the different methods for classifying 3's vs. 5's from the USPS image database. The plots are arranged as in Figure 4.

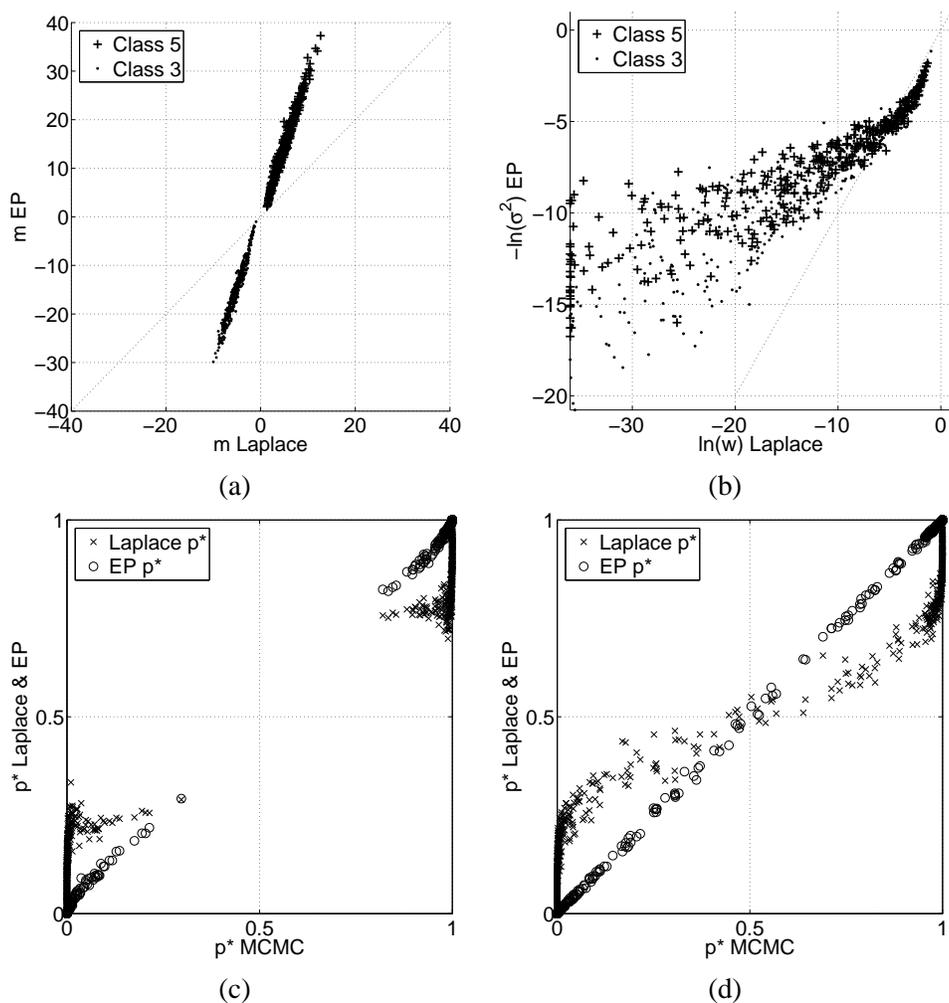


Figure 6: Comparison of approximations $q(\mathbf{f}|\mathcal{D}, \theta) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})$ for a given value of θ . Panel (a) shows a comparison of the means \mathbf{m}_i . In Panel (b) we compare the elements of the diagonal matrices \mathbf{W}_{ii} and Σ_{ii} . Panels (c) and (d) compare predictions p^* obtained by MCMC (abscissa) to predictions obtained from Laplace's method and EP (ordinate). Panel (c) shows predictions on training cases and (d) shows predictions on test cases.

see that the magnitude of the means from the Laplace approximation is much smaller than from EP. The relation appears to be roughly linear. In Figure 6(b) we compare the elements of \mathbf{W} and Σ^{-1} which cause the difference in the approximations (13) and (18) of the posterior covariance matrix \mathbf{A} . We observe that the relatively large entries in \mathbf{W} are larger than the corresponding entries in Σ^{-1} , but in total \mathbf{W} contains more small values than Σ^{-1} . The exact effect on the posterior covariance is difficult to characterise due to the inversion, but intuitively the smaller the values the more the posterior covariance will be similar to the prior.

Figures 6(c-d) compare the predictive uncertainty p^* resulting from the respective approximations to MCMC predictions. For both training and test set we observe that EP and MCMC agree very well, while Laplace’s method shows over-conservative predictions.

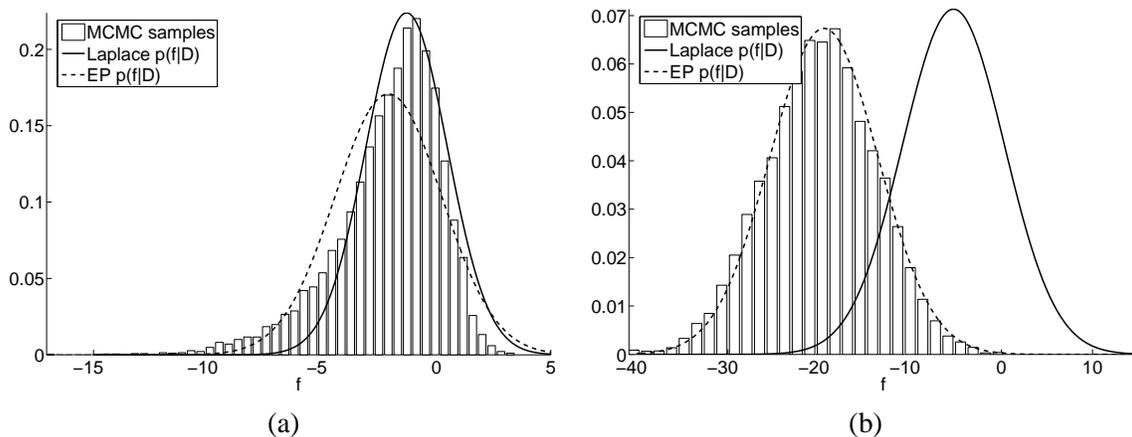


Figure 7: Two marginal distributions $p(f_i|\mathcal{D},\theta)$ from the posterior. For Panel (a) we picked the f_i for which the posterior marginal is maximally skewed (see again Figure 1). The true posterior is approximated by a normalised histogram of 9000 samples of f_i obtained by MCMC sampling. Panel (b) shows a case where EP and Laplace’s approximation differ significantly.

We now inspect the marginal distributions $p(f_i|\mathcal{D},\theta)$ of single latent function values under the posterior approximation. We use hybrid MCMC to generate 9000 samples from the posterior of \mathbf{f} for the above θ . For Laplace’s method and EP the approximated distribution is $\mathcal{N}(f_i|\mathbf{m}_i,\mathbf{A}_{ii})$ where \mathbf{m} and \mathbf{A} are found by the respective approximation techniques.

In general we observe that the marginal distributions of MCMC samples agree very well with the respective marginal distributions of the EP approximation. This supports the claim made in Section 5 where we argued that the marginal distributions of the posterior can be very similar to Gaussians, even if the posterior is a skew distribution. For Laplace’s approximation we find the mean to be underestimated and the marginal distributions to overlap with zero far more than the EP approximations. Figure 7(a) displays the marginal distribution and its approximations for which the MCMC samples show maximal skewness. Figure 7(b) shows a typical example where the EP approximation agrees very well with the MCMC samples. We show this particular example because under the EP approximation $q(y_i = 1|\mathcal{D},\theta) < 0.1\%$ but Laplace’s approximation gives $q(y_i = 1|\mathcal{D},\theta) \simeq 18\%$.

7.4 Lower Bound Approximation

In the context of sparse EP approximations Seeger (2003) proposed a lower bound on the marginal likelihood. The bound is obtained from the EP approximation of the posterior using Jensen’s in-

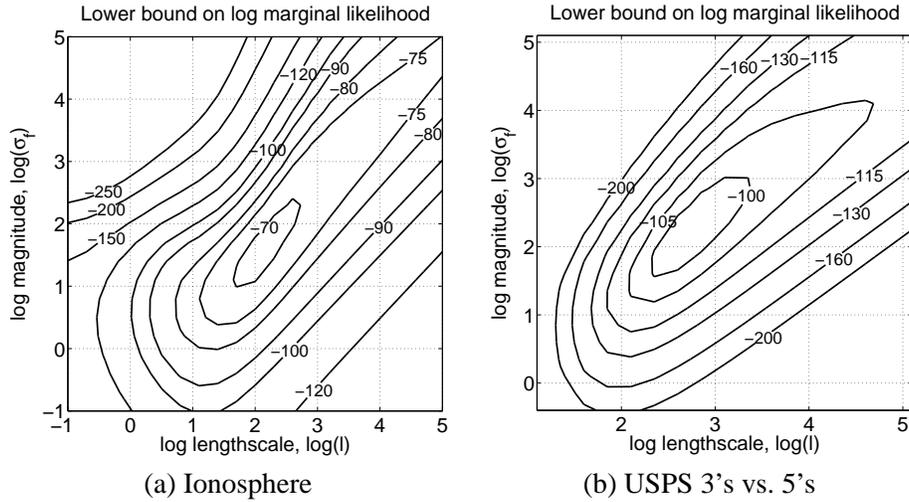


Figure 8: Lower bound on marginal likelihood. Panel (a) shows the lower bound eq. (35) on the marginal likelihood for the Ionosphere data set (compare to left column of Figure 4). Panel (b) shows the value of the lower bound for the USPS 3's vs. 5's (compare to left column of Figure 5)

equality:

$$\ln p(\mathcal{D}|\boldsymbol{\theta}) = \ln \int p(\mathbf{y}|\mathbf{f})\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})d\mathbf{f} \quad (35a)$$

$$\geq \int \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A}) \ln \frac{p(\mathbf{y}|\mathbf{f})\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})}{\mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})}d\mathbf{f} \quad (35b)$$

$$= \sum_{i=1}^m \int \mathcal{N}(f_i|\mathbf{m}_i, \mathbf{A}_{ii}) \ln \Phi(y_i|f_i)df_i \\ - \frac{1}{2}\mathbf{m}^\top \mathbf{K}^{-1}\mathbf{m} - \frac{1}{2}\text{tr}(\mathbf{K}^{-1}\mathbf{A}) + \frac{1}{2}\ln |\mathbf{K}^{-1}\mathbf{A}| + \frac{m}{2}. \quad (35c)$$

Note that the one dimensional integrals in eq. (35c) have to be solved using numerical integration methods.

In sparse EP methods the Gaussian approximation is based on only a subset of observations and so the evidence (27) may be a bad approximation of the total evidence since it does not take all available data into account. Assume that the m points are only a subset of a total of m' observations. The lower bound (35c) can be extended to a lower bound on all m' observations by including all points in the one dimensional integrals over the individual log likelihood terms.

Several authors maximise this lower bound instead of maximising (27) for ML-II hyper-parameter estimation also in the case of non-sparse EP approximations, e.g. Chu and Ghahramani (2005). In Figure 8 we show the value of the lower bound as a function of the hyper-parameters for the Ionosphere and USPS data described in the previous sections (for the full EP approximation). Interestingly, for both data sets the lower bounds appear to be more similar to the approximate evidence obtained by Laplace's method than by EP (compare to the upper left panel in Figures 4 and 5

respectively). However, the maxima of the lower bounds correspond to sub-optimal predictive performances compared to the maxima of the approximate marginal likelihood (27) (compare to the second row in Figures 4 and 5 respectively). Therefore for non-sparse EP approximations the use of (27) seems advisable, which is also computationally advantageous.

7.5 Benchmark Data Sets

In this section we compare the performance of Laplace’s method and Expectation Propagation for GPC on several well known benchmark problems for binary classification.

The *Ionosphere*, the *Wisconsin Breast Cancer*, and the *Sonar* data sets are taken from Hettich et al. (1998). The *Leptograpsus Crabs* and the *Pima Indians Diabetes* data sets were described by Ripley (1996). Note that for the Crabs data set we use the sex (not the colour) of the crabs as target variable. The largest data set in the comparison are the 3’s vs. 5’s from the USPS handwritten digits described above.

We standardise the inputs \mathbf{X} to zero mean and unit variance. All data sets are randomly split into 10 folds of which one at a time is left out as a test set to measure the predictive performance of a model trained (or selected) on the remaining nine folds.

For GPC we implement model selection by ML-II hyper-parameter estimation. We use a conjugate gradient optimisation routine to find a minimum

$$\theta_{\text{ML}} = \underset{\theta}{\operatorname{argmin}} -\ln q(\mathcal{D}|\theta) \quad (36)$$

of the negative log marginal likelihood approximated by Laplace’s method (15) and EP (27) respectively. For the respective θ_{ML} the approximations $\mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})$ are computed and predictions are made for the left out test set. From the predictive distributions the average information (34) is computed and averaged over the ten folds. Furthermore the average error rate E is reported, which equals the average percentage of erroneous class assignments if prediction is understood as a decision problem with symmetric costs (thresholding the predictive uncertainty at $1/2$).

In order to have a better absolute impression of the predictive performance we report the results of support vector machines (SVM) (Schölkopf and Smola, 2002). We use the LIBSVM implementation of C-SVM by Chang and Lin (2001) with a radial basis function kernel which is equivalent to the covariance function (33) up to the signal variance parameter. The values of the length scale parameter ℓ and the regularisation parameter C are found by an *inner loop* of 5-fold cross-validation on the nine training folds respectively. We manually refine the parameter grids and repeat the cross-validation procedure until the performance stabilises.

We use the technique described by Platt (2000) to estimate predictive probabilities from an SVM. This is implemented by fitting a sigmoidal mapping from the unthresholded output of the SVM to the unit interval. The parameters of the mapping are estimated on the test set in the inner loop of 5-fold cross-validation.

Results are summarised in Table 1. Comparing Laplace’s method to EP the latter shows to be more accurate both in terms of error rate and information. While the error rates are relatively similar the predictive distribution obtained by EP shows to be more informative about the test targets. As to be expected by now, the length of the mean vector $\|\mathbf{m}\|$ shows much larger values for the EP approximations. Comparing EP and SVM the results are mixed.

At first sight it may seem surprising that Laplace’s method gives relatively similar error rates compared to EP. Note that for both methods the error rate only depends on the sign of the latent

Data Set	m	n	Laplace			EP			SVM	
			E	I	$\ \mathbf{m}\ $	E	I	$\ \mathbf{m}\ $	E	I
Ionosphere	351	34	8.84	0.591	49.96	7.99	0.661	124.94	5.69	0.681
Wisconsin	683	9	3.21	0.804	62.62	3.21	0.805	84.95	3.21	0.795
Pima Indians	768	8	22.77	0.252	29.05	22.63	0.253	47.49	23.01	0.232
Crabs	200	7	2.0	0.682	112.34	2.0	0.908	2552.97	2.0	0.047
Sonar	208	60	15.36	0.439	26.86	13.85	0.537	15678.55	11.14	0.567
USPS 3 vs 5	1540	256	2.27	0.849	163.05	2.21	0.902	22011.70	2.01	0.918

Table 1: Results for benchmark data sets. The first three columns give the name of the data set, number of observation m and dimension of inputs n . For Laplace’s method and EP the table reports the average error rate E, the average information I (34) and the average length $\|\mathbf{m}\|$ of the mean vector of the Gaussian approximation. For SVMs the error rate and the average information about the test targets are reported.

mean function (5a) at the test locations, which in turn depend on \mathbf{m} only. Therefore the error rate is less sensitive to the accuracy of the approximation to the posterior, but of course depends on the ML-II estimated hyper-parameters, which differ between the methods. Also in the example shown in Figure 3(b) it can be observed that the latent mean functions differ but their sign matches very accurately.

For the Crabs data set all methods show the same error rate but the information content of the predictive distributions differs dramatically. For some test cases the SVM predicts the wrong class with large certainty. Because the mapping of the unthresholded output of the SVM to the predictive probability is estimated from a left out set, the mapping can be poor if too few errors are observed on this.

8. Conclusions

Our experiments reveal serious differences between Laplace’s method and EP when used in GPC models. The results corroborate the considerations about the two approximations based on the structure of the posterior given in Section 5. Although only a handful of data sets have been used in the study, we believe the conclusions to be well-founded and generally valid.

From the structural properties of the posterior we described why Laplace’s method systematically underestimates the mean \mathbf{m} . The resulting approximate posterior GP over latent functions will have too small amplitude, although the sign of the mean function will be mostly correct. As an effect Laplace’s method gives over-conservative predictive probabilities, and diminished information about the test labels. This effect has been shown empirically on several real world examples. Large resulting discrepancies in the actual posterior probabilities were found, even at the training locations, which renders the predictive class probabilities produced under this approximation grossly inaccurate. Note, the difference becomes less dramatic if we only consider the classification error rates obtained by thresholding p^* at $1/2$. For this particular task, we have seen the sign of the latent function tends to be correct (at least at the training locations). However, the performance on benchmark data sets also revealed the error rates obtained by Laplace’s method to be inferior to EP results.

The EP approximation has shown to give results very close to MCMC both in terms of predictive distributions and marginal likelihood estimates. We have shown and explained why the marginal distributions of the posterior can be well approximated by Gaussians.

Further, the marginal likelihood values obtained by Laplace’s method and EP differ systematically which will lead to different results of ML-II hyper-parameter estimation. The discrepancies are similar for different tasks. We were able to exemplify that the EP approximation of the marginal likelihood is accurate. To show this we described how AIS can be used to obtain unbiased estimates of the marginal likelihood for Gaussian process models.

In the experiments summarised in Table 1 we compared the predictive accuracy of GPC to support vector machines. While the SVMs show a tendency to give lower error rates, the information contained in predictive distributions seems comparable. Conceptually GPC comes with the advantage that the Bayesian model selection can be used to set hyper-parameters by ML-II estimation, while the parameters of an SVM usually have to be set by cross-validation (gradient based methods exist, see e.g. Chapelle et al. (2002)).

In summary, we found that EP is the method of choice for approximate inference in binary GPC models, when the computational cost of MCMC is prohibitive. Very good agreement is achieved for both predictive probabilities and marginal likelihood estimates. In contrast, the Laplace approximation is so inaccurate that we advise against its use, especially when predictive probabilities are to be taken seriously.

Acknowledgments

Both authors acknowledge support by the German Research Foundation (DFG) through grant RA 1030/1. We would like to thank Tobias Pfingsten, Jeremy Hill and Matthias Seeger for comments and discussions.

Appendix A. Implementation of Laplace’s Approximation

In Sections 3 we described Laplace’s method for approximate inference in the GPC model and sketched the corresponding computations in Algorithm 1. In this appendix we describe our implementation of the method in more detail. See also the appendices of Williams and Barber (1998).

Computing Laplace’s approximation $\mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{A})$ for given θ the main computational effort is involved in finding the maximum of the unnormalised log posterior $\ln Q$ (eq. (8)). Our implementation uses Newton’s method to find the mode. In each Newton step the vector \mathbf{f} is updated according to

$$\mathbf{f}^{+1} = \mathbf{f} - (\nabla\nabla_{\mathbf{f}}\ln Q(\mathbf{f}'))^{-1}\nabla_{\mathbf{f}}\ln Q(\mathbf{f}') \tag{37a}$$

$$= (\mathbf{K}^{-1} + \mathbf{W})^{-1}(\mathbf{W}\mathbf{f}' + \nabla_{\mathbf{f}}\ln \mathcal{L}(\mathbf{f}')) \tag{37b}$$

until convergence of \mathbf{f} to the mode \mathbf{m} . To ensure convergence the update is accepted if the value of the target function increases, otherwise the step size is shortened until $\ln Q(\mathbf{f}^{+1}) > \ln Q(\mathbf{f}')$.

Computationally Newton’s method is dominated by the repeated inversion of the Hessian. Since \mathbf{K} can be poorly conditioned we use the identity

$$(\mathbf{K}^{-1} + \mathbf{W})^{-1} = \mathbf{K} - \mathbf{K}\mathbf{W}^{\frac{1}{2}}(\mathbf{I} + \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})^{-1}\mathbf{W}^{\frac{1}{2}}\mathbf{K} \tag{38}$$

such that only the well conditioned, positive definite matrix $(\mathbf{I} + \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})$ has to be inverted. In our implementation the inverse is computed from a Cholesky decomposition of this matrix. Note that \mathbf{W} is a diagonal matrix with positive entries, so computing $\mathbf{W}^{\frac{1}{2}}$ is trivial.

Note that implementing the Newton updates (37) only requires the *product* of the inverse Hessian times the gradient which can be computed more efficiently using an iterative conjugate gradient method (Golub and Van Loan, 1989, ch. 10).

Having found the mode \mathbf{m} the marginal likelihood approximation (15) and its derivatives can be computed. The approximate marginal likelihood takes the form

$$\ln q(\mathcal{D}|\boldsymbol{\theta}) = \ln Q(\mathbf{m}) + \frac{m}{2} \ln(2\pi) + \frac{1}{2} \ln |\mathbf{A}| \quad (39a)$$

$$= \ln \mathcal{L}(\mathbf{m}) - \frac{1}{2} \mathbf{m}^\top \mathbf{K}^{-1} \mathbf{m} - \frac{1}{2} \ln |\mathbf{I} + \mathbf{K}\mathbf{W}|. \quad (39b)$$

To avoid the direct inversion of \mathbf{K} in the second term of (39b) we use the recurrence relation (37b). Let $\mathbf{a} = \mathbf{K}^{-1} \mathbf{m}$ then by substituting (38) into (37b) we obtain:

$$\mathbf{a} = (\mathbf{I} - \mathbf{W}^{\frac{1}{2}}(\mathbf{I} + \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}})^{-1}\mathbf{W}^{\frac{1}{2}}\mathbf{K})(\mathbf{W}\mathbf{m} + \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{m})) \quad (40)$$

such that $\mathbf{m}^\top \mathbf{K}^{-1} \mathbf{m} = \mathbf{m}^\top \mathbf{a}$. The determinant in eq. (39b) can be rewritten

$$\ln |\mathbf{I} + \mathbf{K}\mathbf{W}| = \ln |\mathbf{I} + \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}}| \quad (41)$$

and computed from the Cholesky decomposition, that was used to calculate the inverse in eq. (38). Note that if $\mathbf{M} = \mathbf{L}\mathbf{L}^\top$ is a Cholesky decomposition then $\ln |\mathbf{M}| = 2 \sum \ln L_{ii}$.

During ML-II estimation (36) of hyper-parameters the approximate log marginal likelihood (39) is maximised as a function of $\boldsymbol{\theta}$. Our implementation is based on a conjugate gradient optimisation routine such that we also need to compute the derivatives of (39b) with respect to the elements of $\boldsymbol{\theta}$.

The dependency of the approximate marginal likelihood on $\boldsymbol{\theta}$ is two-fold:

$$\frac{\partial \ln q(\mathcal{D}|\boldsymbol{\theta})}{\partial \theta_i} = \sum_{k,l} \frac{\partial \ln q(\mathcal{D}|\boldsymbol{\theta})}{\partial \mathbf{K}_{kl}} \frac{\partial \mathbf{K}_{kl}}{\partial \theta_i} + \frac{\partial \ln q(\mathcal{D}|\boldsymbol{\theta})}{\partial \mathbf{m}^\top} \frac{\partial \mathbf{m}}{\partial \theta_i} \quad (42)$$

there is a direct dependency via the terms involving \mathbf{K} and an implicit dependency through the change in \mathbf{m} (see also Williams and Barber (1998, Appendix B)).

The explicit derivative of eq. (39b) due to the direct dependency of the covariance matrix is

$$\sum_{k,l} \frac{\partial \ln q(\mathcal{D}|\boldsymbol{\theta})}{\partial \mathbf{K}_{kl}} \frac{\partial \mathbf{K}_{kl}}{\partial \theta_i} = \frac{1}{2} \mathbf{m}^\top \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_i} \mathbf{K}^{-1} \mathbf{m} - \frac{1}{2} \text{tr} \left((\mathbf{I} + \mathbf{K}\mathbf{W})^{-1} \frac{\partial \mathbf{K}}{\partial \theta_i} \mathbf{W} \right) \quad (43)$$

where the first term is computed using \mathbf{a} (40) and the inverse in the second term can be rewritten as

$$(\mathbf{I} + \mathbf{K}\mathbf{W})^{-1} = \mathbf{I} - (\mathbf{K}^{-1} + \mathbf{W})^{-1} \mathbf{W} \quad (44)$$

where the inverse (38) is already known.

The implicit derivative accounts for the dependency of eq. (39b) on $\boldsymbol{\theta}$ due to change in the mode \mathbf{m} . Differentiating eq. (39a) with respect to \mathbf{m} reduces to $\partial \ln |\mathbf{A}| / \partial \mathbf{m}$ since \mathbf{m} is the maximum of $\ln Q$ and therefore $\partial \ln Q / \partial \mathbf{m}$ vanishes.

$$\frac{\partial \ln q(\mathcal{D}|\boldsymbol{\theta})}{\partial \mathbf{m}^\top} \frac{\partial \mathbf{m}}{\partial \theta_i} = -\frac{1}{2} \frac{\partial |\mathbf{K}^{-1} + \mathbf{W}|}{\partial \mathbf{m}^\top} \frac{\partial \mathbf{m}}{\partial \theta_i} = -\frac{1}{2} (\mathbf{K}^{-1} + \mathbf{W})^{-1} \frac{\partial \mathbf{W}}{\partial \mathbf{m}^\top} \frac{\partial \mathbf{m}}{\partial \theta_i} \quad (45)$$

The dependency of \mathbf{m} on $\boldsymbol{\theta}_i$ is obtained by differentiating (11a) at \mathbf{m} :

$$0 = \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{m}) - \mathbf{K}^{-1} \mathbf{m} \implies \mathbf{m} = \mathbf{K} \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{m}) \quad (46)$$

so

$$\frac{\partial \mathbf{m}}{\partial \boldsymbol{\theta}_i} = \frac{\partial \mathbf{K}}{\partial \boldsymbol{\theta}_i} \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{m}) + \mathbf{K} \nabla \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{m}) \frac{\partial \mathbf{m}}{\partial \boldsymbol{\theta}_i} = (\mathbf{I} + \mathbf{K} \mathbf{W})^{-1} \frac{\partial \mathbf{K}}{\partial \boldsymbol{\theta}_i} \nabla_{\mathbf{f}} \ln \mathcal{L}(\mathbf{m}) \quad (47)$$

and we have both terms necessary to compute the gradient (42).

To compute the predictive probability $p_* = p(y_* = 1 | \mathbf{x}_*)$ for a test input \mathbf{x}_* the predictive distribution (5) of the latent function value is $\mathcal{N}(f_* | \mu_*, \sigma_*^2)$ where

$$\mu_* = \mathbf{k}_*^{\top} \mathbf{K}^{-1} \mathbf{m} = \mathbf{k}_*^{\top} \mathbf{a} \quad (48a)$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^{\top} \mathbf{W}^{\frac{1}{2}} (\mathbf{I} + \mathbf{W}^{\frac{1}{2}} \mathbf{K} \mathbf{W}^{\frac{1}{2}})^{-1} \mathbf{W}^{\frac{1}{2}} \mathbf{k}_* \quad (48b)$$

and p_* can be computed from eq. (6).

Due to the Cholesky decomposition in (38) computing Laplace's approximation is $O(m^3)$. However, following the implementation we described in this section a Cholesky decomposition has to be computed once per Newton step and all other quantities can be computed from it in at most $O(m^2)$. The number of Newton steps necessary depends on the convergence criterion, the initialisation of \mathbf{f} and the hyper-parameters $\boldsymbol{\theta}$.

Appendix B. Implementation of Expectation Propagation

In this appendix we describe details of our implementation of EP as described in Section 4 and summarised in Algorithm 2. See also the appendices of Seeger (2003).

In our implementation the site functions (17) are parameterised in terms of natural parameters σ_i^{-2} and $\sigma_i^{-2} \mu_i$. For given $\boldsymbol{\theta}$ the algorithm starts by initialising $\mathbf{A} = \mathbf{K}$ and $\sigma_i^{-2} = 0$ and $\sigma_i^{-2} \mu_i = 0$. The algorithm proceeds by updating the site parameters in random order. In each sweep every site function is updated following equations (23), (25), and (26). After each update of a site function the effect on \mathbf{m} and \mathbf{A} has to be computed according to (18). The change in \mathbf{A} can be computed using a rank one update. Let δ be the change in σ_i^{-2} due to the update and \mathbf{e}_i the vector whose i th entry is 1 and all other 0. The relation

$$(\mathbf{K}^{-1} + \boldsymbol{\Sigma}^{-1} + \delta \mathbf{e}_i \mathbf{e}_i^{\top})^{-1} = \mathbf{A} - \mathbf{A} \mathbf{e}_i (\mathbf{A}_{ii} + \delta)^{-1} \mathbf{e}_i^{\top} \mathbf{A} \quad (49)$$

can be used to update \mathbf{A} . Each single update is $O(m^2)$ and repeated m times per sweep, such that the EP algorithm is $O(m^3)$ in time. Because of accumulating numerical errors, after a complete sweep over all site functions we recompute the matrix \mathbf{A} from scratch. For numerical stability we rewrite

$$\mathbf{A} = (\mathbf{K}^{-1} + \boldsymbol{\Sigma}^{-1})^{-1} = \mathbf{K} - \mathbf{K} \boldsymbol{\Sigma}^{-\frac{1}{2}} (\mathbf{I} + \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{K} \boldsymbol{\Sigma}^{-\frac{1}{2}})^{-1} \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{K} \quad (50)$$

and compute the inverse from the Cholesky decomposition of $(\mathbf{I} + \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{K} \boldsymbol{\Sigma}^{-\frac{1}{2}})$.

After convergence the approximate log marginal likelihood (27) can be computed and its partial derivatives with respect to the hyper-parameters:

$$\frac{\partial \ln q(\mathcal{D} | \boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} = -\frac{1}{2} \text{tr} \left(\frac{\partial \mathbf{K}}{\partial \boldsymbol{\theta}_i} \left((\mathbf{K} + \boldsymbol{\Sigma})^{-1} - (\mathbf{K} + \boldsymbol{\Sigma})^{-1} \boldsymbol{\mu} \boldsymbol{\mu}^{\top} (\mathbf{K} + \boldsymbol{\Sigma})^{-1} \right) \right). \quad (51)$$

which do not depend on the Z_i (Seeger, 2005).

The inverse of $\mathbf{K} + \Sigma$ can be computed from the inverse in eq. (50):

$$(\mathbf{K} + \Sigma)^{-1} = \Sigma^{-\frac{1}{2}}(\mathbf{I} + \Sigma^{-\frac{1}{2}}\mathbf{K}\Sigma^{-\frac{1}{2}})^{-1}\Sigma^{-\frac{1}{2}}. \quad (52)$$

For computing the log marginal likelihood (27) also the determinant $|\mathbf{K} + \Sigma|$ has to be computed. By rewriting

$$\ln|\mathbf{K} + \Sigma| = \ln(|\Sigma||\mathbf{I} + \Sigma^{-1}\mathbf{K}|) = \ln|\Sigma| + \ln|\mathbf{I} + \Sigma^{-\frac{1}{2}}\mathbf{K}\Sigma^{-\frac{1}{2}}| \quad (53)$$

we obtain an expression in which the first term is a determinant of a diagonal matrix and the second term can be computed from the Cholesky decomposition that was used to compute the inverse in eq. (50).

To compute the predictive probability $p_* = p(y_* = 1|\mathbf{x}_*)$ for a test input \mathbf{x}_* the predictive distribution (5) of the latent function value is $\mathcal{N}(f_*|\mu_*, \sigma_*^2)$ where

$$\mu_* = \mathbf{k}_*^\top (\mathbf{K} + \Sigma)^{-1} \boldsymbol{\mu} \quad (54a)$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K} + \Sigma)^{-1} \mathbf{k}_* \quad (54b)$$

and p_* can be computed from eq. (6).

The EP algorithm is of computational complexity $O(m^3)$ due to the computations for updating \mathbf{A} . However, per sweep the computation of \mathbf{A} (50) and the m rank one updates sum to more computational effort compared to Laplace's method.

Using a covariance function of the form (33) for some data sets we observed numerical problems during ML-II hyper-parameter estimation because the optimisation algorithm asked to evaluate the marginal likelihood for extremely large signal variances σ^2 . The problem stems from the property that for large values of σ^2 the marginal likelihood becomes insensitive to changes in σ^2 . At this point it is recommended to take another look at Figure 1(b). Intuitively, for large signal variances the prior becomes more spread, such that the likelihood becomes more and more similar to a hard truncation. The marginal likelihood equals the probability mass of the prior in the orthant that is left after truncation. But the probability mass in any of the orthants remains constant if only the signal variance is changed for fixed correlation structure. This argument is based on the assumption that the likelihood implements a hard truncation, which is only an approximation, but this approximation becomes better the larger σ^2 is. Note that this insensitivity of the marginal likelihood with respect to changes in the signal variance can already be observed in the upper parts of the marginal likelihood plots for EP in Figures 4 and 5. A possible solution to this problem is to limit $\sigma^2 < 10^5$, say, since we wouldn't typically expect any new interesting behaviour beyond this.

References

- P. Abrahamsen. A review of Gaussian random fields and correlation functions. Technical Report 917, Norwegian Computing Center, Oslo, 1997.
- C.-C. Chang and C.-J. Lin. *LIBSVM: A library for Support Vector Machines*, 2001. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.
- W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.
- L. Csató and M. Opper. Sparse online Gaussian processes. *Neural Computation*, 14(2):641–669, 2002.
- S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.
- A. Gelman and X.-L. Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185, 1998.
- M. N. Gibbs and D. J. C. MacKay. Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464, 2000.
- G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, second edition, 1989.
- S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.
- N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616, Cambridge, MA, 2003. The MIT Press.
- J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, New York, 2001.
- D. J. C. MacKay. Comparison of approximate methods for handling hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.
- D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003.
- T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2001.
- R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

- R. M. Neal. Regression and classification using Gaussian process priors. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 6*, pages 475–501. Oxford University Press, 1998.
- R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001.
- A. O’Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society, Series B*, 40(1):1–42, 1978.
- M. Opper and O. Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.
- J. C. Platt. Probabilities for SV machines. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–73. The MIT Press, Cambridge, MA, 2000.
- C. E. Rasmussen and C. K. I Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006. *In press*.
- B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.
- M. Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002.
- M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003.
- M. Seeger. Expectation propagation for exponential families, 2005. Note obtainable from <http://www.kyb.tuebingen.mpg.de/~seeger>.
- C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

Clustering with Bregman Divergences

Arindam Banerjee

Srujana Merugu

*Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712, USA.*

ABANERJE@ECE.UTEXAS.EDU

MERUGU@ECE.UTEXAS.EDU

Inderjit S. Dhillon

*Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA*

INDERJIT@CS.UTEXAS.EDU

Joydeep Ghosh

*Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712, USA.*

GHOSH@ECE.UTEXAS.EDU

Editor: John Lafferty

Abstract

A wide variety of distortion functions, such as squared Euclidean distance, Mahalanobis distance, Itakura-Saito distance and relative entropy, have been used for clustering. In this paper, we propose and analyze parametric hard and soft clustering algorithms based on a large class of distortion functions known as Bregman divergences. The proposed algorithms unify centroid-based parametric clustering approaches, such as classical kmeans, the Linde-Buzo-Gray (LBG) algorithm and information-theoretic clustering, which arise by special choices of the Bregman divergence. The algorithms maintain the simplicity and scalability of the classical kmeans algorithm, while generalizing the method to a large class of clustering loss functions. This is achieved by first posing the hard clustering problem in terms of minimizing the loss in Bregman information, a quantity motivated by rate distortion theory, and then deriving an iterative algorithm that monotonically decreases this loss. In addition, we show that there is a bijection between regular exponential families and a large class of Bregman divergences, that we call regular Bregman divergences. This result enables the development of an alternative interpretation of an efficient EM scheme for learning mixtures of exponential family distributions, and leads to a simple soft clustering algorithm for regular Bregman divergences. Finally, we discuss the connection between rate distortion theory and Bregman clustering and present an information theoretic analysis of Bregman clustering algorithms in terms of a trade-off between compression and loss in Bregman information.

Keywords: clustering, Bregman divergences, Bregman information, exponential families, expectation maximization, information theory

1. Introduction

Data clustering is a fundamental “unsupervised” learning procedure that has been extensively studied across varied disciplines over several decades (Jain and Dubes, 1988). Most of the existing parametric clustering methods partition the data into a pre-specified number of partitions with a

cluster representative corresponding to every cluster, such that a well-defined cost function involving the data and the representatives is minimized. Typically, these clustering methods come in two flavors: *hard* and *soft*. In hard clustering, one obtains a disjoint partitioning of the data such that each data point belongs to exactly one of the partitions. In soft clustering, each data point has a certain probability of belonging to each of the partitions. One can think of hard clustering as a special case of soft clustering where these probabilities only take values 0 or 1. The popularity of parametric clustering algorithms stems from their simplicity and scalability.

Several algorithms for solving particular versions of parametric clustering problems have been developed over the years. Among the hard clustering algorithms, the most well-known is the iterative relocation scheme for the Euclidean kmeans algorithm (MacQueen, 1967; Jain and Dubes, 1988; Duda et al., 2001). Another widely used clustering algorithm with a similar scheme is the Linde-Buzo-Gray (LBG) algorithm (Linde et al., 1980; Buzo et al., 1980) based on the Itakura-Saito distance, which has been used in the signal-processing community for clustering speech data. The recently proposed information theoretic clustering algorithm (Dhillon et al., 2003) for clustering probability distributions also has a similar flavor.

The observation that for certain distortion functions, e.g., squared Euclidean distance, KL-divergence (Dhillon et al., 2003), Itakura-Saito distance (Buzo et al., 1980) etc., the clustering problem can be solved using appropriate kmeans type iterative relocation schemes leads to a natural question: *what class of distortion functions admit such an iterative relocation scheme where a global objective function based on the distortion with respect to cluster centroids¹ is progressively decreased?* In this paper, we provide an answer to this question: we show that *such a scheme works for arbitrary Bregman divergences*. In fact, it can be shown (Banerjee et al., 2005) that such a simple scheme works *only* when the distortion is a Bregman divergence. The scope of this result is vast since Bregman divergences include a large number of useful loss functions such as squared loss, KL-divergence, logistic loss, Mahalanobis distance, Itakura-Saito distance, I-divergence, etc.

We pose the hard clustering problem as one of obtaining an optimal quantization in terms of minimizing the loss in *Bregman information*, a quantity motivated by rate distortion theory. A simple analysis then yields a version of the loss function that readily suggests a natural algorithm to solve the clustering problem for arbitrary Bregman divergences. Partitional hard clustering to minimize the loss in *mutual information*, otherwise known as information theoretic clustering (Dhillon et al., 2003), is seen to be a special case of our approach. Thus, this paper unifies several parametric partitional clustering approaches.

Several researchers have observed relationships between Bregman divergences and exponential families (Azoury and Warmuth, 2001; Collins et al., 2001). In this paper, we formally prove an observation made by Forster and Warmuth (2000) that *there exists a unique Bregman divergence corresponding to every regular exponential family*. In fact, we show that there is a bijection between regular exponential families and a class of Bregman divergences, that we call regular Bregman divergences. We show that, with proper representation, the bijection provides an alternative interpretation of a well known efficient EM scheme (Redner and Walker, 1984) for learning mixture models of exponential family distributions. This scheme simplifies the computationally intensive maximization step of the EM algorithm, resulting in a general soft-clustering algorithm for all regular Bregman divergences. We also present an information theoretic analysis of Bregman clustering algorithms in terms of a trade-off between compression and loss in Bregman information.

1. We use the term “cluster centroid” to denote the expectation of the data points in that cluster.

1.1 Contributions

We briefly summarize the main contributions of this paper:

1. In the context of hard clustering, we introduce the concept of *Bregman Information* (Section 3) that measures the minimum expected loss incurred by encoding a set of data points using a constant, where loss is measured in terms of a Bregman divergence. Variance and mutual information are shown to be special cases of Bregman information. Further, we show a close connection between Bregman information and Jensen’s inequality.
2. Hard clustering with Bregman divergences is posed as a quantization problem that involves minimizing loss of Bregman information. We show (Theorem 1 in Section 3) that for any given clustering, the loss in Bregman information is equal to the expected Bregman divergence of data points to their respective cluster centroids. Hence, minimizing either of these quantities yields the same optimal clustering.
3. Based on our analysis of the Bregman clustering problem, we present a meta hard clustering algorithm that is applicable to *all* Bregman divergences (Section 3). The meta clustering algorithm retains the simplicity and scalability of `kmeans` and is a direct generalization of all previously known centroid-based parametric hard clustering algorithms.
4. To obtain a similar generalization for the soft clustering case, we show (Theorem 4, Section 4) that there is a uniquely determined Bregman divergence corresponding to every regular exponential family. This result formally proves an observation made by Forster and Warmuth (2000). In particular, in Section 4.3, we show that the log-likelihood of any parametric exponential family is equal to the negative of the corresponding Bregman divergence to the expectation parameter, up to a fixed additive non-parametric function. Further, in Section 4.4, we define regular Bregman divergences using exponentially convex functions and show that there is a bijection between regular exponential families and regular Bregman divergences.
5. Using the correspondence between exponential families and Bregman divergences, we show that the mixture estimation problem based on regular exponential families is identical to a Bregman soft clustering problem (Section 5). Further, we describe an EM scheme to efficiently solve the mixture estimation problem. Although this particular scheme for learning mixtures of exponential families was previously known (Redner and Walker, 1984), the Bregman divergence viewpoint explaining the efficiency is new. In particular, we give a correctness proof of the efficient M -step updates using properties of Bregman divergences.
6. Finally, we study the relationship between Bregman clustering and rate distortion theory (Section 6). Based on the results in Banerjee et al. (2004a), we observe that the Bregman hard and soft clustering formulations correspond to the “scalar” and asymptotic rate distortion problems respectively, where distortion is measured using a regular Bregman divergence. Further, we show how each of these problems can be interpreted as a trade-off between compression and loss in Bregman information. The information-bottleneck method (Tishby et al., 1999) can be readily derived as a special case of this trade-off.

A word about the notation: bold faced variables, e.g., $\mathbf{x}, \boldsymbol{\mu}$, are used to represent vectors. Sets are represented by calligraphic upper-case alphabets, e.g., \mathcal{X}, \mathcal{Y} . Random variables are represented

by upper-case alphabets, e.g., X, Y . The symbols $\mathbb{R}, \mathbb{N}, \mathbb{Z}$ and \mathbb{R}^d denote the set of reals, the set of natural numbers, the set of integers and the d -dimensional real vector space respectively. Further, \mathbb{R}_+ and \mathbb{R}_{++} denote the set of non-negative and positive real numbers. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\|\mathbf{x}\|$ denotes the L_2 norm, and $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the inner product. Unless otherwise mentioned, \log will represent the natural logarithm. Probability density functions (with respect to the Lebesgue or the counting measure) are denoted by lower case alphabets such as p, q . If a random variable X is distributed according to ν , expectation of functions of X are denoted by $E_X[\cdot]$, or by $E_\nu[\cdot]$ when the random variable is clear from the context. The interior, relative interior, boundary, closure and closed convex hull of a set \mathcal{X} are denoted by $\text{int}(\mathcal{X})$, $\text{ri}(\mathcal{X})$, $\text{bd}(\mathcal{X})$, $\text{cl}(\mathcal{X})$ and $\text{co}(\mathcal{X})$ respectively. The effective domain of a function f , i.e., set of all x such that $f(x) < +\infty$ is denoted by $\text{dom}(f)$ while the range is denoted by $\text{range}(f)$. The inverse of a function f , when well-defined, is denoted by f^{-1} .

2. Preliminaries

In this section, we define the Bregman divergence corresponding to a strictly convex function and present some examples.

Definition 1 (Bregman, 1967; Censor and Zenios, 1998) Let $\phi : \mathcal{S} \mapsto \mathbb{R}$, $\mathcal{S} = \text{dom}(\phi)$ be a strictly convex function defined on a convex set $\mathcal{S} \subseteq \mathbb{R}^d$ such that ϕ is differentiable on $\text{ri}(\mathcal{S})$, assumed to be nonempty. The *Bregman divergence* $d_\phi : \mathcal{S} \times \text{ri}(\mathcal{S}) \mapsto [0, \infty)$ is defined as

$$d_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle,$$

where $\nabla\phi(\mathbf{y})$ represents the gradient vector of ϕ evaluated at \mathbf{y} .

Example 1 Squared Euclidean distance is perhaps the simplest and most widely used Bregman divergence. The underlying function $\phi(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x} \rangle$ is strictly convex, differentiable on \mathbb{R}^d and

$$\begin{aligned} d_\phi(\mathbf{x}, \mathbf{y}) &= \langle \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{y}, \mathbf{y} \rangle - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle \\ &= \langle \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{y}, \mathbf{y} \rangle - \langle \mathbf{x} - \mathbf{y}, 2\mathbf{y} \rangle \\ &= \langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle = \|\mathbf{x} - \mathbf{y}\|^2. \end{aligned}$$

Example 2 Another widely used Bregman divergence is the KL-divergence. If \mathbf{p} is a discrete probability distribution so that $\sum_{j=1}^d p_j = 1$, the negative entropy $\phi(\mathbf{p}) = \sum_{j=1}^d p_j \log_2 p_j$ is a convex function. The corresponding Bregman divergence is

$$\begin{aligned} d_\phi(\mathbf{p}, \mathbf{q}) &= \sum_{j=1}^d p_j \log_2 p_j - \sum_{j=1}^d q_j \log_2 q_j - \langle \mathbf{p} - \mathbf{q}, \nabla\phi(\mathbf{q}) \rangle \\ &= \sum_{j=1}^d p_j \log_2 p_j - \sum_{j=1}^d q_j \log_2 q_j - \sum_{j=1}^d (p_j - q_j)(\log_2 q_j + \log_2 e) \\ &= \sum_{j=1}^d p_j \log_2 \left(\frac{p_j}{q_j} \right) - \log_2 e \sum_{j=1}^d (p_j - q_j) \\ &= KL(\mathbf{p} \parallel \mathbf{q}), \end{aligned}$$

the KL-divergence between the two distributions as $\sum_{j=1}^d q_j = \sum_{j=1}^d p_j = 1$.

Table 1: Bregman divergences generated from some convex functions.

Domain	$\phi(\mathbf{x})$	$d_\phi(\mathbf{x}, \mathbf{y})$	Divergence
\mathbb{R}	x^2	$(x - y)^2$	Squared loss
\mathbb{R}_+	$x \log x$	$x \log(\frac{x}{y}) - (x - y)$	
$[0, 1]$	$x \log x + (1 - x) \log(1 - x)$	$x \log(\frac{x}{y}) + (1 - x) \log(\frac{1-x}{1-y})$	Logistic loss ³
\mathbb{R}_{++}	$-\log x$	$\frac{x}{y} - \log(\frac{x}{y}) - 1$	Itakura-Saito distance
\mathbb{R}	e^x	$e^x - e^y - (x - y)e^y$	
\mathbb{R}^d	$\ \mathbf{x}\ ^2$	$\ \mathbf{x} - \mathbf{y}\ ^2$	Squared Euclidean distance
\mathbb{R}^d	$\mathbf{x}^T A \mathbf{x}$	$(\mathbf{x} - \mathbf{y})^T A (\mathbf{x} - \mathbf{y})$	Mahalanobis distance ⁴
d -Simplex	$\sum_{j=1}^d x_j \log_2 x_j$	$\sum_{j=1}^d x_j \log_2(\frac{x_j}{y_j})$	KL-divergence
\mathbb{R}_+^d	$\sum_{j=1}^d x_j \log x_j$	$\sum_{j=1}^d x_j \log(\frac{x_j}{y_j}) - \sum_{j=1}^d (x_j - y_j)$	Generalized I-divergence

Example 3 Itakura-Saito distance is another Bregman divergence that is widely used in signal processing. If $F(e^{j\theta})$ is the power spectrum² of a signal $f(t)$, then the functional $\phi(F) = -\frac{1}{2\pi} \int_{-\pi}^{\pi} \log(F(e^{j\theta})) d\theta$ is convex in F and corresponds to the negative entropy rate of the signal assuming it was generated by a stationary Gaussian process (Palus, 1997; Cover and Thomas, 1991). The Bregman divergence between $F(e^{j\theta})$ and $G(e^{j\theta})$ (the power spectrum of another signal $g(t)$) is given by

$$\begin{aligned} d_\phi(F, G) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(-\log(F(e^{j\theta})) + \log(G(e^{j\theta})) - (F(e^{j\theta}) - G(e^{j\theta})) \left(-\frac{1}{G(e^{j\theta})} \right) \right) d\theta \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(-\log\left(\frac{F(e^{j\theta})}{G(e^{j\theta})}\right) + \frac{F(e^{j\theta})}{G(e^{j\theta})} - 1 \right) d\theta, \end{aligned}$$

which is exactly the Itakura-Saito distance between the power spectra $F(e^{j\theta})$ and $G(e^{j\theta})$ and can also be interpreted as the I-divergence (Csiszár, 1991) between the generating processes under the assumption that they are equal mean, stationary Gaussian processes (Kazakos and Kazakos, 1980).

Table 1 contains a list of some common convex functions and their corresponding Bregman divergences. Bregman divergences have several interesting and useful properties, such as non-negativity, convexity in the first argument, etc. For details see Appendix A.

3. Bregman Hard Clustering

In this section, we introduce a new concept called the Bregman information of a random variable based on ideas from Shannon’s rate distortion theory. Then, we motivate the Bregman hard clustering problem as a quantization problem that involves minimizing the loss in Bregman information and show its equivalence to a more direct formulation, i.e., the problem of finding a partitioning and a representative for each of the partitions such that the expected Bregman divergence of the data

2. Note that $F(\cdot)$ is a function and it is possible to extend the notion of Bregman divergences to the space of functions (Csiszár, 1995; Grünwald and Dawid, 2004).
3. For $x \in \{0, 1\}$ (Bernoulli) and $y \in (0, 1)$ (posterior probability for $x = 1$), we have $x \log(\frac{x}{y}) + (1 - x) \log(\frac{1-x}{1-y}) = \log(1 + \exp(-f(x)g(y)))$, i.e., the logistic loss with $f(x) = 2x - 1$ and $g(y) = \log(\frac{y}{1-y})$.
4. The matrix A is assumed to be positive definite; $(\mathbf{x} - \mathbf{y})^T A (\mathbf{x} - \mathbf{y})$ is called the Mahalanobis distance when A is the inverse of the covariance matrix.

points from their representatives is minimized. We then present a clustering algorithm that generalizes the iterative relocation scheme of kmeans to monotonically decrease the loss in Bregman information.

3.1 Bregman Information

The dual formulation of Shannon's celebrated rate distortion problem (Cover and Thomas, 1991; Grünwald and Vitányi, 2003) involves finding a coding scheme with a given rate, i.e., average number of bits per symbol, such that the expected distortion between the source random variable and the decoded random variable is minimized. The achieved distortion is called the *distortion rate function*, i.e., the infimum distortion achievable for a given rate. Now consider a random variable X that takes values in a finite set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ (\mathcal{S} is convex) following a discrete probability measure ν . Let the distortion be measured by a Bregman divergence d_ϕ . Consider a simple encoding scheme that represents the random variable by a constant vector \mathbf{s} , i.e., codebook size is one, or rate is zero. The solution to the rate-distortion problem in this case is the trivial assignment. The corresponding distortion-rate function is given by $E_\nu[d_\phi(X, \mathbf{s})]$ that depends on the choice of the representative \mathbf{s} and can be optimized by picking the right representative. We call this optimal distortion-rate function the *Bregman information* of the random variable X for the Bregman divergence d_ϕ and denote it by $I_\phi(X)$, i.e.,

$$I_\phi(X) = \min_{\mathbf{s} \in \text{ri}(\mathcal{S})} E_\nu[d_\phi(X, \mathbf{s})] = \min_{\mathbf{s} \in \text{ri}(\mathcal{S})} \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \mathbf{s}). \quad (1)$$

The optimal vector \mathbf{s} that achieves the minimal distortion will be called the *Bregman representative* or, simply the *representative* of X . The following theorem states that this representative always exists, is uniquely determined and, surprisingly, *does not depend* on the choice of Bregman divergence. In fact, the minimizer is just the expectation of the random variable X .

Proposition 1 *Let X be a random variable that take values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ following a positive probability measure ν such that $E_\nu[X] \in \text{ri}(\mathcal{S})$.⁵ Given a Bregman divergence $d_\phi : \mathcal{S} \times \text{ri}(\mathcal{S}) \mapsto [0, \infty)$, the problem*

$$\min_{\mathbf{s} \in \text{ri}(\mathcal{S})} E_\nu[d_\phi(X, \mathbf{s})] \quad (2)$$

has a unique minimizer given by $\mathbf{s}^\dagger = \boldsymbol{\mu} = E_\nu[X]$.

Proof The function we are trying to minimize is $J_\phi(\mathbf{s}) = E_\nu[d_\phi(X, \mathbf{s})] = \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \mathbf{s})$. Since $\boldsymbol{\mu} = E_\nu[X] \in \text{ri}(\mathcal{S})$, the objective function is well-defined at $\boldsymbol{\mu}$. Now, $\forall \mathbf{s} \in \text{ri}(\mathcal{S})$,

$$\begin{aligned} J_\phi(\mathbf{s}) - J_\phi(\boldsymbol{\mu}) &= \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \mathbf{s}) - \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) \\ &= \phi(\boldsymbol{\mu}) - \phi(\mathbf{s}) - \left\langle \sum_{i=1}^n \nu_i \mathbf{x}_i - \mathbf{s}, \nabla \phi(\mathbf{s}) \right\rangle + \left\langle \sum_{i=1}^n \nu_i \mathbf{x}_i - \boldsymbol{\mu}, \nabla \phi(\boldsymbol{\mu}) \right\rangle \\ &= \phi(\boldsymbol{\mu}) - \phi(\mathbf{s}) - \langle \boldsymbol{\mu} - \mathbf{s}, \nabla \phi(\mathbf{s}) \rangle \\ &= d_\phi(\boldsymbol{\mu}, \mathbf{s}) \geq 0, \end{aligned}$$

5. The assumption that $E_\nu[X] \in \text{ri}(\mathcal{S})$ is not restrictive since a violation can occur only when $\text{co}(\mathcal{X}) \subset \text{bd}(\mathcal{S})$, i.e., the entire convex hull of \mathcal{X} is on the boundary of \mathcal{S} .

with equality only when $\mathbf{s} = \boldsymbol{\mu}$ by the strict convexity of ϕ (Appendix A, Property 1). Hence, $\boldsymbol{\mu}$ is the unique minimizer of J_ϕ . \blacksquare

Note that the minimization in (2) is with respect to the second argument of d_ϕ . Proposition 1 is somewhat surprising since Bregman divergences are not necessarily convex in the second argument as the following example demonstrates.

Example 4 Consider $\phi(\mathbf{x}) = \sum_{j=1}^3 x_j^3$ defined on \mathbb{R}_+^3 so that $d_\phi(\mathbf{x}, \mathbf{s}) = \sum_{j=1}^3 (x_j^3 - s_j^3 - 3(x_j - s_j)s_j^2)$. For the random variable X distributed uniformly over the set $\mathcal{X} = \{(1, 1, 1), (2, 2, 2), (3, 3, 3), (4, 4, 4), (5, 5, 5)\}$,

$$E[d_\phi(X, \mathbf{s})] = 135 + 2 \sum_{j=1}^3 s_j^3 - 9 \sum_{j=1}^3 s_j^2,$$

which is clearly not convex in \mathbf{s} since the Hessian $\nabla^2 J_\phi(\mathbf{s}) = \text{diag}(12\mathbf{s} - 18)$ is not positive definite. However, $J_\phi(\mathbf{s})$ is uniquely minimized by $\mathbf{s} = (3, 3, 3)$, i.e., the expectation of the random variable X .

Interestingly, the converse of Proposition 1 is also true, i.e., for all random variables X , if $E[X]$ minimizes the expected distortion of X to a fixed point for a smooth distortion function $F(x, y)$ (see Appendix B for details), then $F(x, y)$ has to be a Bregman divergence (Banerjee et al., 2005). Thus, Bregman divergences are *exhaustive* with respect to the property proved in Proposition 1.

Using Proposition 1, we can now give a more direct definition of Bregman information as follows:

Definition 2 Let X be a random variable that takes values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathcal{S}$ following a probability measure ν . Let $\boldsymbol{\mu} = E_\nu[X] = \sum_{i=1}^n \nu_i \mathbf{x}_i \in \text{ri}(\mathcal{S})$ and let $d_\phi : \mathcal{S} \times \text{ri}(\mathcal{S}) \mapsto [0, \infty)$ be a Bregman divergence. Then the *Bregman Information* of X in terms of d_ϕ is defined as

$$I_\phi(X) = E_\nu[d_\phi(X, \boldsymbol{\mu})] = \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}).$$

Example 5 (Variance) Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ be a set in \mathbb{R}^d , and consider the uniform measure, i.e., $\nu_i = \frac{1}{n}$, over \mathcal{X} . The Bregman information of X with squared Euclidean distance as the Bregman divergence is given by

$$I_\phi(X) = \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \boldsymbol{\mu}\|^2,$$

which is just the sample variance.

Example 6 (Mutual Information) By definition, the mutual information $I(U; V)$ between two discrete random variables U and V with joint distribution $\{p(u_i, v_j)\}_{i=1}^n \}_{j=1}^m$ is given by

$$\begin{aligned} I(U; V) &= \sum_{i=1}^n \sum_{j=1}^m p(u_i, v_j) \log \frac{p(u_i, v_j)}{p(u_i)p(v_j)} = \sum_{i=1}^n p(u_i) \sum_{j=1}^m p(v_j|u_i) \log \frac{p(v_j|u_i)}{p(v_j)} \\ &= \sum_{i=1}^n p(u_i) KL(p(V|u_i) \parallel p(V)). \end{aligned}$$

Consider a random variable Z_u that takes values in the set of probability distributions $\mathcal{Z}_u = \{p(V|u_i)\}_{i=1}^n$ following the probability measure $\{v_i\}_{i=1}^n = \{p(u_i)\}_{i=1}^n$ over this set. The mean (distribution) of Z_u is given by

$$\boldsymbol{\mu} = E_v[p(V|u)] = \sum_{i=1}^n p(u_i)p(V|u_i) = \sum_{i=1}^n p(u_i, V) = p(V) .$$

Hence,

$$I(U;V) = \sum_{i=1}^n v_i d_\phi(p(V|u_i), \boldsymbol{\mu}) = I_\phi(Z_u) ,$$

i.e., mutual information is the Bregman information of Z_u when d_ϕ is the KL-divergence. Similarly, for a random variable Z_v that takes values in the set of probability distributions $\mathcal{Z}_v = \{p(U|v_j)\}_{j=1}^m$ following the probability measure $\{v_j\}_{j=1}^m = \{p(v_j)\}_{j=1}^m$ over this set, one can show that $I(U;V) = I_\phi(Z_v)$. The Bregman information of Z_u and Z_v can also be interpreted as the Jensen-Shannon divergence of the sets \mathcal{Z}_u and \mathcal{Z}_v (Dhillon et al., 2003).

Example 7 The Bregman information corresponding to Itakura-Saito distance also has a useful interpretation. Let $\mathcal{F} = \{F_i\}_{i=1}^n$ be a set of power spectra corresponding to n different signals, and let v be a probability measure on \mathcal{F} . Then, the Bregman information of a random variable F that takes values in \mathcal{F} following v , with Itakura-Saito distance as the Bregman divergence, is given by

$$\begin{aligned} I_\phi(F) &= \sum_{i=1}^n v_i d_\phi(F_i, \bar{F}) = \sum_{i=1}^n \frac{v_i}{2\pi} \int_{-\pi}^{\pi} \left(-\log \left(\frac{F_i(e^{j\theta})}{\bar{F}(e^{j\theta})} \right) + \frac{F_i(e^{j\theta})}{\bar{F}(e^{j\theta})} - 1 \right) d\theta \\ &= -\frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{i=1}^n v_i \log \left(\frac{F_i(e^{j\theta})}{\bar{F}(e^{j\theta})} \right) d\theta, \end{aligned}$$

where \bar{F} is the marginal average power spectrum. Based on the connection between the corresponding convex function ϕ and the negative entropy of Gaussian processes (Cover and Thomas, 1991; Palus, 1997), it can be shown that the Bregman information $I_\phi(F)$ is the Jensen-Shannon divergence of the generating processes under the assumption that they are equal mean, stationary Gaussian processes. Further, consider a n -class signal classification problem where each class of signals is assumed to be generated by a certain Gaussian process. Now, if $P_e(t)$ is the optimal Bayes error for this classification problem averaged upto time t , then $P_e(t)$ is bounded above and below by functions of the Chernoff coefficient $B(t)$ (Kazakos and Kazakos, 1980) of the generating Gaussian processes. The asymptotic value of this Chernoff coefficient as t tends to ∞ is a function of the Bregman information of F , i.e.,

$$\lim_{t \rightarrow \infty} B(t) = \exp\left(-\frac{1}{2}I_\phi(F)\right).$$

and is directly proportional to the optimal Bayes error.

3.1.1 JENSEN'S INEQUALITY AND BREGMAN INFORMATION

An alternative interpretation of Bregman information can also be made in terms of Jensen's inequality (Cover and Thomas, 1991). Given any convex function ϕ , for any random variable X , Jensen's inequality states that

$$E[\phi(X)] \geq \phi(E[X]) .$$

A direct calculation using the definition of Bregman information shows that (Banerjee et al., 2004b)

$$\begin{aligned} E[\phi(X)] - \phi(E[X]) &\stackrel{(a)}{=} E[\phi(X)] - \phi(E[X]) - E[\langle X - E[X], \nabla\phi(E[X]) \rangle] \\ &\stackrel{(b)}{=} E[\phi(X) - \phi(E[X]) - \langle X - E[X], \nabla\phi(E[X]) \rangle] \\ &= E[d_\phi(X, E[X])] = I_\phi(X) \geq 0, \end{aligned}$$

where (a) follows since the last term is 0, and (b) follows from the linearity of expectation. Thus the difference between the two sides of Jensen's inequality is exactly equal to the Bregman information.

3.2 Clustering Formulation

Let X be a random variable that takes values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ following the probability measure ν . When X has a large Bregman information, it may not suffice to encode X using a single representative since a lower quantization error may be desired. In such a situation, a natural goal is to split the set \mathcal{X} into k disjoint partitions $\{\mathcal{X}_h\}_{h=1}^k$, each with its own Bregman representative, such that a random variable M over the partition representatives serves as an appropriate quantization of X . Let $\mathcal{M} = \{\boldsymbol{\mu}_h\}_{h=1}^k$ denote the set of representatives, and $\pi = \{\pi_h\}_{h=1}^k$ with $\pi_h = \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i$ denote the induced probability measure on \mathcal{M} . Then the induced random variable M takes values in \mathcal{M} following π .

The quality of the quantization M can be measured by the expected Bregman divergence between X and M , i.e., $E_{X,M}[d_\phi(X, M)]$. Since M is a deterministic function of X , the expectation is actually over the distribution of X , so that

$$E_X[d_\phi(X, M)] = \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) = \sum_{h=1}^k \pi_h \sum_{\mathbf{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) = E_\pi[I_\phi(X_h)],$$

where X_h is the random variable that takes values in the partition \mathcal{X}_h following a probability distribution $\frac{\nu_i}{\pi_h}$, and $I_\phi(X_h)$ is the Bregman information of X_h . Thus, the quality of the quantization is equal to the expected Bregman information of the partitions.

An alternative way of measuring the quality of the quantization M can be formulated from an information theoretic viewpoint. In information-theoretic clustering (Dhillon et al., 2003), the quality of the partitioning is measured in terms of the loss in mutual information resulting from the quantization of the original random variable X . Extending this formulation, we can measure the quality of the quantization M by the loss in Bregman information due to the quantization, i.e., by $I_\phi(X) - I_\phi(M)$. For $k = n$, the best choice is of course $M = X$ with no loss in Bregman information. For $k = 1$, the best quantization is to pick $E_\nu[X]$ with probability 1, incurring a loss of $I_\phi(X)$. For intermediate values of k , the solution is less obvious.

Interestingly the two possible formulations outlined above turn out to be identical (see Theorem 1 below). We choose the information theoretic viewpoint to pose the problem, since we will study the connections of both the hard and soft clustering problems to rate distortion theory in Section 6. Thus we define the *Bregman hard clustering problem* as that of finding a partitioning of \mathcal{X} , or, equivalently, finding the random variable M , such that *the loss in Bregman information due to quantization, $L_\phi(M) = I_\phi(X) - I_\phi(M)$, is minimized*. Typically, clustering algorithms assume a uniform measure, i.e., $\nu_i = \frac{1}{n}, \forall i$, over the data, which is clearly a special case of our formulation.

The following theorem shows that the loss in Bregman information and the expected Bregman information of the partitions are equal.

Theorem 1 Let X be a random variable that takes values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$ following the positive probability measure ν . Let $\{\mathcal{X}_h\}_{h=1}^k$ be a partitioning of \mathcal{X} and let $\pi_h = \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i$ be the induced measure π on the partitions. Let X_h be the random variable that takes values in \mathcal{X}_h following $\frac{\nu_i}{\pi_h}$ for $\mathbf{x}_i \in \mathcal{X}_h$, for $h = 1, \dots, k$. Let $\mathcal{M} = \{\boldsymbol{\mu}_h\}_{h=1}^k$ with $\boldsymbol{\mu}_h \in \text{ri}(\mathcal{S})$ denote the set of representatives of $\{\mathcal{X}_h\}_{h=1}^k$, and M be a random variable that takes values in \mathcal{M} following π . Then,

$$L_\phi(M) = I_\phi(X) - I_\phi(M) = E_\pi[I_\phi(X_h)] = \sum_{h=1}^k \pi_h \sum_{\mathbf{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h).$$

Proof By definition,

$$\begin{aligned} I_\phi(X) &= \sum_{i=1}^n \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) = \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) \\ &= \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i \{ \phi(\mathbf{x}_i) - \phi(\boldsymbol{\mu}) - \langle \mathbf{x}_i - \boldsymbol{\mu}, \nabla \phi(\boldsymbol{\mu}) \rangle \} \\ &= \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i \{ \phi(\mathbf{x}_i) - \phi(\boldsymbol{\mu}_h) - \langle \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla \phi(\boldsymbol{\mu}_h) \rangle + \langle \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla \phi(\boldsymbol{\mu}_h) \rangle \\ &\quad + \phi(\boldsymbol{\mu}_h) - \phi(\boldsymbol{\mu}) - \langle \mathbf{x}_i - \boldsymbol{\mu}_h + \boldsymbol{\mu}_h - \boldsymbol{\mu}, \nabla \phi(\boldsymbol{\mu}) \rangle \} \\ &= \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i \{ d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) + d_\phi(\boldsymbol{\mu}_h, \boldsymbol{\mu}) + \langle \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla \phi(\boldsymbol{\mu}_h) - \nabla \phi(\boldsymbol{\mu}) \rangle \} \\ &= \sum_{h=1}^k \pi_h \sum_{\mathbf{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) + \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\boldsymbol{\mu}_h, \boldsymbol{\mu}) \\ &\quad + \sum_{h=1}^k \pi_h \sum_{\mathbf{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} \langle \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla \phi(\boldsymbol{\mu}_h) - \nabla \phi(\boldsymbol{\mu}) \rangle \\ &= \sum_{h=1}^k \pi_h I_\phi(X_h) + \sum_{h=1}^k \pi_h d_\phi(\boldsymbol{\mu}_h, \boldsymbol{\mu}) + \sum_{h=1}^k \pi_h \left\langle \sum_{\mathbf{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} \mathbf{x}_i - \boldsymbol{\mu}_h, \nabla \phi(\boldsymbol{\mu}_h) - \nabla \phi(\boldsymbol{\mu}) \right\rangle \\ &= E_\pi[I_\phi(X_h)] + I_\phi(M), \end{aligned}$$

since $\sum_{\mathbf{x}_i \in \mathcal{X}_h} \frac{\nu_i}{\pi_h} \mathbf{x}_i = \boldsymbol{\mu}_h$. ■

Note that $I_\phi(X)$ can be interpreted as the “total Bregman information”, and $I_\phi(M)$ can be interpreted as the “between-cluster Bregman information” since it is a measure of divergence between the cluster representatives, while $L_\phi(M)$ can be interpreted as the “within-cluster Bregman information”. Thus Theorem 1 states that the total Bregman information equals the sum of the within-cluster Bregman information and between-cluster Bregman information. This is a generalization of the corresponding result for squared Euclidean distances (Duda et al., 2001).

Using Theorem 1, the Bregman clustering problem of minimizing the loss in Bregman information can be written as

$$\min_M (I_\phi(X) - I_\phi(M)) = \min_M \left(\sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h) \right). \quad (3)$$

Algorithm 1 Bregman Hard Clustering

Input: Set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset S \subseteq \mathbb{R}^d$, probability measure ν over \mathcal{X} , Bregman divergence $d_\phi : S \times \text{ri}(S) \mapsto \mathbb{R}$, number of clusters k .

Output: \mathcal{M}^\dagger , local minimizer of $L_\phi(\mathcal{M}) = \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h)$ where $\mathcal{M} = \{\boldsymbol{\mu}_h\}_{h=1}^k$, hard partitioning $\{\mathcal{X}_h\}_{h=1}^k$ of \mathcal{X} .

Method:

Initialize $\{\boldsymbol{\mu}_h\}_{h=1}^k$ with $\boldsymbol{\mu}_h \in \text{ri}(S)$ (one possible initialization is to choose $\boldsymbol{\mu}_h \in \text{ri}(S)$ at random)

repeat

{The Assignment Step}

Set $\mathcal{X}_h \leftarrow \emptyset$, $1 \leq h \leq k$

for $i = 1$ to n **do**

$\mathcal{X}_h \leftarrow \mathcal{X}_h \cup \{\mathbf{x}_i\}$

where $h = h^\dagger(\mathbf{x}_i) = \underset{h'}{\operatorname{argmin}} d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_{h'})$

end for

{The Re-estimation Step}

for $h = 1$ to k **do**

$\pi_h \leftarrow \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i$

$\boldsymbol{\mu}_h \leftarrow \frac{1}{\pi_h} \sum_{\mathbf{x}_i \in \mathcal{X}_h} \nu_i \mathbf{x}_i$

end for

until convergence

return $\mathcal{M}^\dagger \leftarrow \{\boldsymbol{\mu}_h\}_{h=1}^k$

Thus, the loss in Bregman information is minimized if the set of representatives \mathcal{M} is such that the expected Bregman divergence of points in the original set \mathcal{X} to their corresponding representatives is minimized. We shall investigate the relationship of this formulation to rate distortion theory in detail in Section 6.

3.3 Clustering Algorithm

The objective function given in (3) suggests a natural iterative relocation algorithm for solving the Bregman hard clustering problem (see Algorithm 1). It is easy to see that classical kmeans, the LBG algorithm (Buzo et al., 1980) and the information theoretic clustering algorithm (Dhillon et al., 2003) are special cases of Bregman hard clustering for squared Euclidean distance, Itakura-Saito distance and KL-divergence respectively. The following propositions prove the convergence of the Bregman hard clustering algorithm.

Proposition 2 *The Bregman hard clustering algorithm (Algorithm 1) monotonically decreases the loss function in (3).*

Proof Let $\{\mathcal{X}_h^{(t)}\}_{h=1}^k$ be the partitioning of \mathcal{X} after the t^{th} iteration and let $\mathcal{M}^{(t)} = \{\boldsymbol{\mu}_h^{(t)}\}_{h=1}^k$ be the corresponding set of cluster representatives. Then,

$$\begin{aligned} L_\phi(M^{(t)}) &= \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(t)}} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h^{(t)}) \stackrel{(a)}{\geq} \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(t)}} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_{h^\dagger(\mathbf{x}_i)}^{(t)}) \\ &\stackrel{(b)}{\geq} \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h^{(t+1)}} \nu_i d_\phi(\mathbf{x}_i, \boldsymbol{\mu}_h^{(t+1)}) = L_\phi(M^{(t+1)}), \end{aligned}$$

where (a) follows from the assignment step, and (b) follows from the re-estimation step and Proposition 1. Note that if equality holds, i.e., if the loss function value is equal at consecutive iterations, then the algorithm will terminate. ■

Proposition 3 *The Bregman hard clustering algorithm (Algorithm 1) terminates in a finite number of steps at a partition that is locally optimal, i.e., the total loss cannot be decreased by either (a) the assignment step or by (b) changing the means of any existing clusters.*

Proof The result follows since the algorithm monotonically decreases the objective function value, and the number of distinct clusterings is finite. ■

In addition to local optimality, the Bregman hard clustering algorithm has the following interesting properties.

Exhaustiveness: The Bregman hard clustering algorithm with cluster centroids as optimal representatives works for *all* Bregman divergences and *only* for Bregman divergences since the arithmetic mean is the best predictor *only* for Bregman divergences (Banerjee et al., 2005). However, it is possible to have a similar alternate minimization based clustering algorithm for distance functions that are not Bregman divergences, the primary difference being that the optimal cluster representative, when it exists, will no longer be the arithmetic mean or the expectation. The `convex-kmeans` clustering algorithm (Modha and Spangler, 2003) and the generalizations of the LBG algorithm (Linde et al., 1980) are examples of such alternate minimization schemes where a (unique) representative exists because of convexity.

Linear Separators: For all Bregman divergences, the partitions induced by the Bregman hard clustering algorithm are separated by hyperplanes. In particular, the locus of points that are equidistant to two fixed points μ_1, μ_2 in terms of a Bregman divergence is given by $\mathcal{X} = \{\mathbf{x} \mid d_\phi(\mathbf{x}, \mu_1) = d_\phi(\mathbf{x}, \mu_2)\}$, i.e., the set of points,

$$\{\mathbf{x} \mid \langle \mathbf{x}, \nabla\phi(\mu_2) - \nabla\phi(\mu_1) \rangle = (\phi(\mu_1) - \langle \mu_1, \nabla\phi(\mu_1) \rangle) - (\phi(\mu_2) - \langle \mu_2, \nabla\phi(\mu_2) \rangle)\},$$

which corresponds to a hyperplane.

Scalability: The computational complexity of each iteration of the Bregman hard clustering algorithm is linear in the number of data points and the number of desired clusters for *all* Bregman divergences, which makes the algorithm scalable and appropriate for large clustering problems.

Applicability to mixed data types: The Bregman hard clustering algorithm is applicable to mixed data types that are commonly encountered in machine learning. One can choose different convex functions that are appropriate and meaningful for different subsets of the features. The Bregman divergence corresponding to a convex combination of the component convex functions can then be used to cluster the data.

4. Relationship with Exponential Families

We now turn our attention to *soft* clustering with Bregman divergences. To accomplish our goal, we first establish that there is a unique Bregman divergence corresponding to every regular exponential

family distribution. Later, we make this relation more precise by establishing a bijection between regular exponential families and *regular Bregman divergences*. The correspondence will be used to develop the Bregman soft clustering algorithm in Section 5. To present our results, we first review some background information on exponential families and Legendre duality in Sections 4.1 and 4.2 respectively.

4.1 Exponential Families

Consider a measurable space (Ω, \mathcal{B}) where \mathcal{B} is a σ -algebra on the set Ω . Let \mathbf{t} be a measurable mapping from Ω to a set $\mathcal{T} \subseteq \mathbb{R}^d$, where \mathcal{T} may be discrete (e.g., $\mathcal{T} \subset \mathbb{N}$). Let $p_0 : \mathcal{T} \mapsto \mathbb{R}_+$ be any function such that if (Ω, \mathcal{B}) is endowed with a measure $dP_0(\omega) = p_0(\mathbf{t}(\omega))d\mathbf{t}(\omega)$, then $\int_{\omega \in \Omega} dP_0(\omega) < \infty$. The measure P_0 is absolutely continuous with respect to the Lebesgue measure $d\mathbf{t}(\omega)$. When \mathcal{T} is a discrete set, $d\mathbf{t}(\omega)$ is the counting measure and P_0 is absolutely continuous with respect to the counting measure.⁶

Now, $\mathbf{t}(\omega)$ is a random variable from $(\Omega, \mathcal{B}, P_0)$ to $(\mathcal{T}, \sigma(\mathcal{T}))$, where $\sigma(\mathcal{T})$ denotes the σ -algebra generated by \mathcal{T} . Let Θ be defined as the set of all parameters $\theta \in \mathbb{R}^d$ for which

$$\int_{\omega \in \Omega} \exp(\langle \theta, \mathbf{t}(\omega) \rangle) dP_0(\omega) < \infty .$$

Based on the definition of Θ , it is possible to define a function $\psi : \Theta \mapsto \mathbb{R}$ such that

$$\psi(\theta) = \log \left(\int_{\omega \in \Omega} \exp(\langle \theta, \mathbf{t}(\omega) \rangle) dP_0(\omega) \right) . \quad (4)$$

A family of probability distributions \mathcal{F}_ψ parameterized by a d -dimensional vector $\theta \in \Theta \subseteq \mathbb{R}^d$ such that the probability density functions with respect to the measure $d\mathbf{t}(\omega)$ can be expressed in the form

$$f(\omega; \theta) = \exp(\langle \theta, \mathbf{t}(\omega) \rangle - \psi(\theta)) p_0(\mathbf{t}(\omega)) \quad (5)$$

is called an *exponential family* with *natural statistic* $\mathbf{t}(\omega)$, *natural parameter* θ and *natural parameter space* Θ . In particular, if the components of $\mathbf{t}(\omega)$ are affinely independent, i.e., \nexists non-zero $\mathbf{a} \in \mathbb{R}^d$ such that $\langle \mathbf{a}, \mathbf{t}(\omega) \rangle = c$ (a constant) $\forall \omega \in \Omega$, then this representation is said to be *minimal*.⁷ For a minimal representation, there exists a unique probability density $f(\omega; \theta)$ for every choice of $\theta \in \Theta$ (Wainwright and Jordan, 2003). \mathcal{F}_ψ is called a *full exponential family* of order d in such a case. In addition, if the parameter space Θ is open, i.e., $\Theta = \text{int}(\Theta)$, then \mathcal{F}_ψ is called a *regular exponential family*.

It can be easily seen that if $\mathbf{x} \in \mathbb{R}^d$ denotes the natural statistic $\mathbf{t}(\omega)$, then the probability density function $g(\mathbf{x}; \theta)$ (with respect to the appropriate measure $d\mathbf{x}$) given by

$$g(\mathbf{x}; \theta) = \exp(\langle \theta, \mathbf{x} \rangle - \psi(\theta)) p_0(\mathbf{x}) \quad (6)$$

is such that $f(\omega; \theta)/g(\mathbf{x}; \theta)$ does not depend on θ . Thus, \mathbf{x} is a sufficient statistic (Amari and Nagaoka, 2001) for the family, and in fact, can be shown (Barndorff-Nielsen, 1978) to be minimally

6. For conciseness, we abuse notation and continue to use the Lebesgue integral sign even for counting measures. The integral in this case actually denotes a sum over \mathcal{T} . Further, the use of absolute continuity in the context of counting measure is non-standard. We say the measure P_0 is absolutely continuous with respect to the counting measure μ_c if $P_0(E) = 0$ for every set with $\mu_c(E) = 0$, where E is a discrete set.

7. Strictly speaking, \nexists non-zero \mathbf{a} such that $P_0(\{\omega : \langle \mathbf{t}(\omega), \mathbf{a} \rangle = c\}) = 1$.

sufficient. For instance, the natural statistic for the one-dimensional Gaussian distributions denoted by $f(\omega; \sigma, \mu) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(\omega-\mu)^2}{2\sigma^2})$ is given by $\mathbf{x} = [\omega, \omega^2]$ and the corresponding natural parameter turns out to be $\boldsymbol{\theta} = [\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}]$, which can be easily verified to be minimally sufficient. For our analysis, it is convenient to work with the minimal natural sufficient statistic \mathbf{x} and hence, we redefine regular exponential families in terms of the probability density of $\mathbf{x} \in \mathbb{R}^d$, noting that the original probability space can actually be quite general.

Definition 3 A multivariate parametric family \mathcal{F}_ψ of distributions $\{p_{(\psi, \boldsymbol{\theta})} | \boldsymbol{\theta} \in \Theta = \text{int}(\Theta) = \text{dom}(\psi) \subseteq \mathbb{R}^d\}$ is called a regular exponential family if each probability density is of the form

$$p_{(\psi, \boldsymbol{\theta})}(\mathbf{x}) = \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta})) p_0(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^d,$$

where \mathbf{x} is a minimal sufficient statistic for the family.

The function $\psi(\boldsymbol{\theta})$ is known as the *log partition function* or the *cumulant function* corresponding to the exponential family. Given a regular exponential family \mathcal{F}_ψ , the log-partition function ψ is uniquely determined up to a constant additive term. It can be shown (Barndorff-Nielsen, 1978) that Θ is a non-empty convex set in \mathbb{R}^d and ψ is a convex function. In fact, it is possible to prove a stronger result that characterizes ψ in terms of a special class of convex functions called Legendre functions, which are defined below.

Definition 4 (Rockafellar (1970)) Let ψ be a proper, closed⁸ convex function with $\Theta = \text{int}(\text{dom}(\psi))$. The pair (Θ, ψ) is called a convex function of Legendre type or a Legendre function if the following properties are satisfied:

- (L1) Θ is non-empty,
- (L2) ψ is strictly convex and differentiable on Θ ,
- (L3) $\forall \boldsymbol{\theta}_b \in \text{bd}(\Theta), \lim_{\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}_b} \|\nabla \psi(\boldsymbol{\theta})\| \rightarrow \infty$ where $\boldsymbol{\theta} \in \Theta$.

Based on this definition, we now state a critical property of the cumulant function of any regular exponential family.

Lemma 1 *Let ψ be the cumulant function of a regular exponential family with natural parameter space $\Theta = \text{dom}(\psi)$. Then ψ is a proper, closed convex function with $\text{int}(\Theta) = \Theta$ and (Θ, ψ) is a convex function of Legendre type.*

The above result directly follows from Theorems 8.2, 9.1 and 9.3 of Barndorff-Nielsen (1978).

4.2 Expectation Parameters and Legendre Duality

Consider a d -dimensional real random vector X distributed according to a regular exponential family density $p_{(\psi, \boldsymbol{\theta})}$ specified by the natural parameter $\boldsymbol{\theta} \in \Theta$. The expectation of X with respect to $p_{(\psi, \boldsymbol{\theta})}$, also called the *expectation parameter*, is given by

$$\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\theta}) = E_{p_{(\psi, \boldsymbol{\theta})}}[X] = \int_{\mathbb{R}^d} \mathbf{x} p_{(\psi, \boldsymbol{\theta})}(\mathbf{x}) d\mathbf{x}. \tag{7}$$

8. A convex function ψ is proper if $\text{dom}(\psi)$ is non-empty and $\forall \mathbf{x} \in \text{dom}(\psi), \psi(\mathbf{x}) > -\infty$. A convex function is closed if it is lower semi-continuous.

It can be shown (Barndorff-Nielsen, 1978; Amari, 1995) that the expectation and natural parameters have a one-one correspondence with each other and span spaces that exhibit a dual relationship. To specify the duality more precisely, we first define conjugate functions.

Definition 5 (Rockafellar (1970)) Let ψ be a real-valued function on \mathbb{R}^d . Then its *conjugate function* ψ^* is given by

$$\psi^*(\mathbf{t}) = \sup_{\boldsymbol{\theta} \in \text{dom}(\psi)} \{\langle \mathbf{t}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta})\}. \quad (8)$$

Further, if ψ is a proper closed convex function, ψ^* is also a proper closed convex function and $\psi^{**} = \psi$.

When ψ is strictly convex and differentiable over $\Theta = \text{int}(\text{dom}(\psi))$, we can obtain the unique $\boldsymbol{\theta}^\dagger$ that corresponds to the supremum in (8) by setting the gradient of $\langle \mathbf{t}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta})$ to zero, i.e.,

$$\nabla(\langle \mathbf{t}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta}))|_{\boldsymbol{\theta}=\boldsymbol{\theta}^\dagger} = 0 \quad \Rightarrow \quad \mathbf{t} = \nabla\psi(\boldsymbol{\theta}^\dagger). \quad (9)$$

The strict convexity of ψ implies that $\nabla\psi$ is monotonic and it is possible to define the inverse function $(\nabla\psi)^{-1} : \Theta^* \mapsto \Theta$, where $\Theta^* = \text{int}(\text{dom}(\psi^*))$. If the pair (Θ, ψ) is of Legendre type, then it can be shown (Rockafellar, 1970) that (Θ^*, ψ^*) is also of Legendre type, and (Θ, ψ) and (Θ^*, ψ^*) are called Legendre duals of each other. Further, the gradient mappings are continuous and form a bijection between the two open sets Θ and Θ^* . The relation between (Θ, ψ) and (Θ^*, ψ^*) result is formally stated below.

Theorem 2 (Rockafellar (1970)) Let ψ be a real-valued proper closed convex function with conjugate function ψ^* . Let $\Theta = \text{int}(\text{dom}(\psi))$ and $\Theta^* = \text{int}(\text{dom}(\psi^*))$. If (Θ, ψ) is a convex function of Legendre type, then

- (i) (Θ^*, ψ^*) is a convex function of Legendre type,
- (ii) (Θ, ψ) and (Θ^*, ψ^*) are Legendre duals of each other,
- (iii) The gradient function $\nabla\psi : \Theta \mapsto \Theta^*$ is a one-to-one function from the open convex set Θ onto the open convex set Θ^* ,
- (iv) The gradient functions $\nabla\psi, \nabla\psi^*$ are continuous, and $\nabla\psi^* = (\nabla\psi)^{-1}$.

Let us now look at the relationship between the natural parameter $\boldsymbol{\theta}$ and the expectation parameter $\boldsymbol{\mu}$ defined in (7). Differentiating the identity $\int p_{(\psi, \boldsymbol{\theta})}(\mathbf{x})d\mathbf{x} = 1$ with respect to $\boldsymbol{\theta}$ gives us $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{\theta}) = \nabla\psi(\boldsymbol{\theta})$, i.e., the expectation parameter $\boldsymbol{\mu}$ is the image of the natural parameter $\boldsymbol{\theta}$ under the gradient mapping $\nabla\psi$. Let ϕ be defined as the conjugate of ψ , i.e.,

$$\phi(\boldsymbol{\mu}) = \sup_{\boldsymbol{\theta} \in \Theta} \{\langle \boldsymbol{\mu}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta})\}. \quad (10)$$

Since (Θ, ψ) is a convex function of Legendre type (Lemma 1), the pairs (Θ, ψ) and $(\text{int}(\text{dom}(\phi)), \phi)$ are Legendre duals of each other from Theorem 2, i.e., $\phi = \psi^*$ and $\text{int}(\text{dom}(\phi)) = \Theta^*$. Thus, the mappings between the dual spaces $\text{int}(\text{dom}(\phi))$ and Θ are given by the Legendre transformation

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \nabla\psi(\boldsymbol{\theta}) \quad \text{and} \quad \boldsymbol{\theta}(\boldsymbol{\mu}) = \nabla\phi(\boldsymbol{\mu}). \quad (11)$$

Further, the conjugate function ϕ can be expressed as

$$\phi(\boldsymbol{\mu}) = \langle \boldsymbol{\theta}(\boldsymbol{\mu}), \boldsymbol{\mu} \rangle - \psi(\boldsymbol{\theta}(\boldsymbol{\mu})), \quad \forall \boldsymbol{\mu} \in \text{int}(\text{dom}(\phi)). \quad (12)$$

4.3 Exponential Families and Bregman Divergences

We are now ready to explicitly state the formal connection between exponential families of distributions and Bregman divergences. It has been observed in the literature that exponential families and Bregman divergences have a close relationship that can be exploited for several learning problems. In particular, Forster and Warmuth (2000)[Section 5.1] remarked that the log-likelihood of the density of an exponential family distribution $p_{(\psi, \theta)}$ can be written as the sum of the negative of a uniquely determined Bregman divergence $d_\phi(\mathbf{x}, \boldsymbol{\mu})$ and a function that does not depend on the distribution parameters. In our notation, this can be written as

$$\log(p_{(\psi, \theta)}(\mathbf{x})) = -d_\phi(\mathbf{x}, \boldsymbol{\mu}(\theta)) + \log(b_\phi(\mathbf{x})) , \quad (13)$$

where ϕ is the conjugate function of ψ and $\boldsymbol{\mu} = \boldsymbol{\mu}(\theta) = \nabla\psi(\theta)$ is the expectation parameter corresponding to θ . The result was later used by Collins et al. (2001) to extend PCA to exponential families. However, as we explain below, a formal proof is required to show that (13) holds for all instances \mathbf{x} of interest. We focus on the case when $p_{(\psi, \theta)}$ is a *regular* exponential family.

To get an intuition of the main result, observe that the log-likelihood of any exponential family, considering only the parametric terms, can be written as

$$\begin{aligned} \langle \mathbf{x}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta}) &= (\langle \boldsymbol{\mu}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta})) + \langle \mathbf{x} - \boldsymbol{\mu}, \boldsymbol{\theta} \rangle \\ &= \phi(\boldsymbol{\mu}) + \langle \mathbf{x} - \boldsymbol{\mu}, \nabla\phi(\boldsymbol{\mu}) \rangle , \end{aligned}$$

from (11) and (12), where $\boldsymbol{\mu} \in \text{int}(\text{dom}(\phi))$. Therefore, for any $\mathbf{x} \in \text{dom}(\phi)$, $\boldsymbol{\theta} \in \Theta$, and $\boldsymbol{\mu} \in \text{int}(\text{dom}(\phi))$, one can write

$$\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta}) = -d_\phi(\mathbf{x}, \boldsymbol{\mu}) + \phi(\mathbf{x}) .$$

Then considering the density of an exponential family with respect to the appropriate measure $d\mathbf{x}$, we have

$$\log(p_{(\psi, \theta)}(\mathbf{x})) = \langle \mathbf{x}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta}) + \log p_0(\mathbf{x}) = -d_\phi(\mathbf{x}, \boldsymbol{\mu}) + \log(b_\phi(\mathbf{x})) ,$$

where $b_\phi(\mathbf{x}) = \exp(\phi(\mathbf{x}))p_0(\mathbf{x})$.

Thus (13) follows directly from Legendre duality for $\mathbf{x} \in \text{dom}(\phi)$. However, for (13) to be useful, one would like to ensure that it is true for all individual ‘‘instances’’ \mathbf{x} that can be drawn following the exponential distribution $p_{(\psi, \theta)}$. Let I_ψ denote the set of such instances. Establishing (13) can be tricky for all $\mathbf{x} \in I_\psi$ since the relationship between I_ψ and $\text{dom}(\phi)$ is not apparent. Further, there are distributions for which the instances space I_ψ and the expectation parameter space $\text{int}(\text{dom}(\phi))$ are disjoint, as the following example shows.

Example 8 A Bernoulli random variable X takes values in $\{0, 1\}$ such that $p(X = 1) = q$ and $p(X = 0) = 1 - q$, for some $q \in [0, 1]$. The instance space for X is just $I_\psi = \{0, 1\}$. The cumulant function for X is $\psi(\theta) = \log(1 + \exp(\theta))$ with $\Theta = \mathbb{R}$ (see Table 2). A simple calculation shows that the conjugate function $\phi(\mu) = \mu \log \mu + (1 - \mu) \log(1 - \mu)$, $\forall \mu \in (0, 1)$. Since ϕ is a closed function, we obtain $\phi(\mu) = 0$ for $\mu \in \{0, 1\}$ by taking limits. Thus, the effective domain of ϕ is $[0, 1]$ and $\mu = q$, whereas the expectation parameter space is given by $\text{int}(\text{dom}(\phi)) = (0, 1)$. Hence the instance space I_ψ and the expectation parameter space $\text{int}(\text{dom}(\phi))$ are disjoint; however $I_\psi \subset \text{dom}(\phi)$.

In this particular case, since the “instances” lie within $\text{dom}(\phi)$, the relation (13) does hold for all $\mathbf{x} \in I_\psi$. However, it remains to be shown that $I_\psi \subseteq \text{dom}(\phi)$ for all regular exponential family distributions.

In order to establish such a result for all regular exponential family distributions, we need to formally define the set of instances I_ψ . If the measure P_0 is absolutely continuous with respect to the counting measure, then $\mathbf{x} \in I_\psi$ if $p_{(\psi, \theta)}(\mathbf{x}) > 0$. On the other hand, if P_0 is absolutely continuous with respect to the Lebesgue measure, then $\mathbf{x} \in I_\psi$ if all sets with positive Lebesgue measure that contain \mathbf{x} have positive probability mass. A closer look reveals that the set of instances I_ψ is independent of the choice of θ . In fact, I_ψ is just the support of P_0 and can be formally defined as follows.

Definition 6 Let I_ψ denote the set of instances that can be drawn following $p_{(\psi, \theta)}(\mathbf{x})$. Then, $\mathbf{x}_0 \in I_\psi$ if $\forall I$ such that $\mathbf{x}_0 \in I$ and $\int_I d\mathbf{x} > 0$, we have $\int_I dP_0(\mathbf{x}) > 0$, where P_0 is as defined in Section 4.1; also see footnote 6.

The following theorem establishes the crucial result that the set of instances I_ψ is always a subset of $\text{dom}(\phi)$.

Theorem 3 Let I_ψ be the set of instances as in Definition 6. Then, $I_\psi \subseteq \text{dom}(\phi)$ where ϕ is the conjugate function of ψ .

The above result follows from Theorem 9.1 and related results in Barndorff-Nielsen (1978). We have included the proof in Appendix C.

We are now ready to formally show that there is a unique Bregman divergence corresponding to every regular exponential family distribution. Note that, by Theorem 3, it is sufficient to establish the relationship for all $\mathbf{x} \in \text{dom}(\phi)$.

Theorem 4 Let $p_{(\psi, \theta)}$ be the probability density function of a regular exponential family distribution. Let ϕ be the conjugate function of ψ so that $(\text{int}(\text{dom}(\phi)), \phi)$ is the Legendre dual of (Θ, ψ) . Let $\theta \in \Theta$ be the natural parameter and $\mu \in \text{int}(\text{dom}(\phi))$ be the corresponding expectation parameter. Let d_ϕ be the Bregman divergence derived from ϕ . Then $p_{(\psi, \theta)}$ can be uniquely expressed as

$$p_{(\psi, \theta)}(\mathbf{x}) = \exp(-d_\phi(\mathbf{x}, \mu))b_\phi(\mathbf{x}), \quad \forall \mathbf{x} \in \text{dom}(\phi) \quad (14)$$

where $b_\phi : \text{dom}(\phi) \mapsto \mathbb{R}_+$ is a uniquely determined function.

Proof For all $\mathbf{x} \in \text{dom}(\phi)$, we have

$$\begin{aligned} p_{(\psi, \theta)}(\mathbf{x}) &= \exp(\langle \mathbf{x}, \theta \rangle - \psi(\theta))p_0(\mathbf{x}) \\ &= \exp(\phi(\mu) + \langle \mathbf{x} - \mu, \nabla\phi(\mu) \rangle)p_0(\mathbf{x}) \quad (\text{using (11) and (12)}) \\ &= \exp(-\{\phi(\mathbf{x}) - \phi(\mu) - \langle \mathbf{x} - \mu, \nabla\phi(\mu) \rangle\} + \phi(\mathbf{x}))p_0(\mathbf{x}) \\ &= \exp(-d_\phi(\mathbf{x}, \mu))b_\phi(\mathbf{x}), \end{aligned}$$

where $b_\phi(\mathbf{x}) = \exp(\phi(\mathbf{x}))p_0(\mathbf{x})$.

We observe that $p_{(\psi, \theta)}$ uniquely determines the log-partition function ψ to a constant additive term so that the gradient space of all the possible functions ψ is the same, i.e., the expectation parameter $\mu = \nabla\psi(\theta)$ corresponding to θ is uniquely determined and the corresponding conjugate functions ϕ differ only by a constant additive term. Hence the Bregman divergence $d_\phi(\mathbf{x}, \mu)$ derived

from any of these conjugate functions will be identical since constant additive terms do not change the corresponding Bregman divergence (Appendix A, Property 4). The Legendre duality between ϕ and ψ also ensures that no two exponential families correspond to the same Bregman divergence, i.e., the mapping is one-to-one. Further, since $p_{(\psi, \theta)}(\mathbf{x})$ is well-defined on $\text{dom}(\phi)$, and the corresponding $d_\phi(\mathbf{x}, \boldsymbol{\mu})$ is unique, the function $b_\phi(\mathbf{x}) = \exp(d_\phi(\mathbf{x}, \boldsymbol{\mu}))p_{(\psi, \theta)}(\mathbf{x})$ is uniquely determined. ■

4.4 Bijection with Regular Bregman Divergences

From Theorem 4 we note that every regular exponential family corresponds to a unique and distinct Bregman divergence (one-to-one mapping). Now, we investigate whether there is a regular exponential family corresponding to every choice of Bregman divergence (onto mapping).

For regular exponential families, the cumulant function ψ as well as its conjugate ϕ are convex functions of Legendre type. Hence, for a Bregman divergence generated from a convex function ϕ to correspond to a regular exponential family, it is necessary that ϕ be of Legendre type. Further, it is necessary that the Legendre conjugate ψ of ϕ to be C^∞ , since cumulant functions of regular exponential families are C^∞ . However, it is not clear if these conditions are sufficient. Instead, we provide a sufficiency condition using exponentially convex functions (Akhizer, 1965; Ehm et al., 2003), which are defined below.

Definition 7 A function $f : \Theta \mapsto \mathbb{R}_{++}$, $\Theta \subseteq \mathbb{R}^d$ is called exponentially convex if the kernel $K_f(\alpha, \beta) = f(\alpha + \beta)$, with $\alpha + \beta \in \Theta$, satisfies

$$\sum_{i=1}^n \sum_{j=1}^n K_f(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) u_i \bar{u}_j \geq 0$$

for any set $\{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n\} \subseteq \Theta$ with $\boldsymbol{\theta}_i + \boldsymbol{\theta}_j \in \Theta$, $\forall i, j$, and $\{u_1, \dots, u_n\} \subset \mathbb{C}$ (\bar{u}_j denotes the complex conjugate of u_j), i.e, the kernel K_f is positive semi-definite.

Although it is well known that the logarithm of an exponentially convex function is a convex function (Akhizer, 1965), we are interested in the case where the logarithm is strictly convex with an open domain. Using this class of exponentially convex functions, we now define a class of Bregman divergences called *regular Bregman divergences*.

Definition 8 Let $f : \Theta \mapsto \mathbb{R}_{++}$ be a continuous exponentially convex function such that Θ is open and $\psi(\boldsymbol{\theta}) = \log(f(\boldsymbol{\theta}))$ is strictly convex. Let ϕ be the conjugate function of ψ . Then we say that the Bregman divergence d_ϕ derived from ϕ is a *regular Bregman divergence*.

We will now prove that there is a bijection between regular exponential families and regular Bregman divergences. The crux of the argument relies on results in harmonic analysis connecting positive definiteness to integral transforms (Berg et al., 1984). In particular, we use a result due to Devinatz (1955) that relates exponentially convex functions to Laplace transforms of bounded non-negative measures.

Theorem 5 (Devinatz (1955)) Let $\Theta \subseteq \mathbb{R}^d$ be an open convex set. A necessary and sufficient condition that there exists a unique, bounded, non-negative measure ν such that $f : \Theta \mapsto \mathbb{R}_{++}$ can be

represented as

$$f(\boldsymbol{\theta}) = \int_{\mathbf{x} \in \mathbb{R}^d} \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle) d\nu(\mathbf{x}) \quad (15)$$

is that f is continuous and exponentially convex.

We also need the following result to establish the bijection.

Lemma 2 *Let ψ be the cumulant of an exponential family with base measure P_0 and natural parameter space $\Theta \subseteq \mathbb{R}^d$. Then, if P_0 is concentrated on an affine subspace of \mathbb{R}^d then ψ is not strictly convex.*

Proof Let $P_0(\mathbf{x})$ be concentrated on an affine subspace $S = \{\mathbf{x} \in \mathbb{R}^d \mid \langle \mathbf{x}, \mathbf{b} \rangle = c\}$ for some $\mathbf{b} \in \mathbb{R}^d$ and $c \in \mathbb{R}$. Let $I = \{\boldsymbol{\theta} \mid \boldsymbol{\theta} = \alpha \mathbf{b}, \alpha \in \mathbb{R}\}$. Then, for any $\boldsymbol{\theta} = \alpha \mathbf{b} \in I$, we have $\langle \mathbf{x}, \boldsymbol{\theta} \rangle = \alpha c \forall \mathbf{x} \in S$ and the cumulant is given by

$$\begin{aligned} \psi(\boldsymbol{\theta}) &= \log \left(\int_{\mathbf{x} \in \mathbb{R}^d} \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle) dP_0(\mathbf{x}) \right) = \log \left(\int_{\mathbf{x} \in S} \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle) dP_0(\mathbf{x}) \right) \\ &= \log \left(\int_{\mathbf{x} \in S} \exp(\alpha c) dP_0(\mathbf{x}) \right) = \log(\exp(\alpha c) P_0(S)) = \alpha c + \log(P_0(S)) \\ &= \langle \mathbf{x}_0, \boldsymbol{\theta} \rangle + \log(P_0(S)), \end{aligned}$$

for any $\mathbf{x}_0 \in S$, implying that ψ is not strictly convex. ■

There are two parts to the proof leading to the bijection result. Note that we have already established in Theorem 4 that there is a unique Bregman divergence corresponding to every exponential family distribution. In the first part of the proof, we show that these Bregman divergences are regular (one-to-one). Then we show that there exists a unique regular exponential family determined by every regular Bregman divergence (onto).

Theorem 6 *There is a bijection between regular exponential families and regular Bregman divergences.*

Proof First we prove the ‘one-to-one’ part, i.e., there is a regular Bregman divergence corresponding to every regular exponential family F_ψ with cumulant function ψ and natural parameter space Θ . Since F_ψ is a regular exponential family, there exists a non-negative bounded measure ν such that for all $\boldsymbol{\theta} \in \Theta$,

$$\begin{aligned} 1 &= \int_{\mathbf{x} \in \mathbb{R}^d} \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta})) d\nu(\mathbf{x}) \\ \Rightarrow \exp(\psi(\boldsymbol{\theta})) &= \int_{\mathbf{x} \in \mathbb{R}^d} \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle) d\nu(\mathbf{x}). \end{aligned}$$

Thus, from Theorem 5, $\exp(\psi(\boldsymbol{\theta}))$ is a continuous exponentially convex function with the open set Θ as its domain. Further, being the cumulant of a regular exponential family, ψ is strictly convex. Therefore, the Bregman divergence d_ϕ derived from the conjugate function ϕ of ψ is a regular Bregman divergence.

Next we prove the ‘onto’ part, i.e., every regular Bregman divergence corresponds to a unique regular exponential family. Let the regular Bregman divergence d_ϕ be generated by ϕ and let ψ be

the conjugate of ϕ . Since d_ϕ is a regular Bregman divergence, by Definition 8, ψ is strictly convex with $\text{dom}(\psi) = \Theta$ being an open set. Further, the function $\exp(\psi(\boldsymbol{\theta}))$ is a continuous, exponentially convex function. From Theorem 5, there exists a unique non-negative bounded measure ν that satisfies (15). Since Θ is non-empty, we can choose some fixed $\mathbf{b} \in \Theta$ so that

$$\exp(\psi(\mathbf{b})) = \int_{\mathbf{x} \in \mathbb{R}^d} \exp(\langle \mathbf{x}, \mathbf{b} \rangle) d\nu(\mathbf{x})$$

and so $dP_0(\mathbf{x}) = \exp(\langle \mathbf{x}, \mathbf{b} \rangle - \psi(\mathbf{b})) d\nu(\mathbf{x})$ is a probability density function. The set of all $\boldsymbol{\theta} \in \mathbb{R}^d$ for which

$$\int_{\mathbf{x} \in \mathbb{R}^d} \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle) dP_0(\mathbf{x}) < \infty$$

is same as the set $\{\boldsymbol{\theta} \in \mathbb{R}^d \mid \exp(\psi(\boldsymbol{\theta} + \mathbf{b}) - \psi(\mathbf{b})) < \infty\} = \{\boldsymbol{\theta} \in \mathbb{R}^d \mid \boldsymbol{\theta} + \mathbf{b} \in \Theta\}$ which is just a translated version of Θ itself. For any $\boldsymbol{\theta}$ such that $\boldsymbol{\theta} + \mathbf{b} \in \Theta$, we have

$$\int_{\mathbf{x} \in \mathbb{R}^d} \exp(\langle \mathbf{x}, \boldsymbol{\theta} + \mathbf{b} \rangle - \psi(\boldsymbol{\theta} + \mathbf{b})) d\nu(\mathbf{x}) = 1 .$$

Hence, the exponential family \mathcal{F}_ψ consisting of densities of the form

$$p_{(\psi, \boldsymbol{\theta})}(\mathbf{x}) = \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta}))$$

with respect to the measure ν has Θ as its natural parameter space and $\psi(\boldsymbol{\theta})$ as the cumulant function.

Since ψ is strictly convex on Θ , it follows from Lemma 2 that the measure P_0 is not concentrated in an affine subspace of \mathbb{R}^d , i.e., \mathbf{x} is a minimal statistic for \mathcal{F}_ψ . Therefore, the exponential family generated by P_0 and \mathbf{x} is full. Since Θ is also open, it follows that \mathcal{F}_ψ is a regular exponential family.

Finally we show that the family is unique. Since only d_ϕ is given, the generating convex function could be $\bar{\phi}(\mathbf{x}) = \phi(\mathbf{x}) + \langle \mathbf{x}, \mathbf{a} \rangle + c$ for $\mathbf{a} \in \mathbb{R}^d$ and a constant $c \in \mathbb{R}$. The corresponding conjugate function $\bar{\psi}(\boldsymbol{\theta}) = \psi(\boldsymbol{\theta} - \mathbf{a}) - c$ differs from ψ only by a constant. Hence, the exponential family is exactly \mathcal{F}_ψ . That completes the proof. ■

4.5 Examples

Table 2 shows the various functions of interest for some popular exponential families. We now look at two of these distributions in detail and obtain the corresponding Bregman divergences.

Example 9 The most well-known exponential family is that of Gaussian distributions, in particular uniform variance, spherical Gaussian distributions with densities of the form

$$p(\mathbf{x}; \mathbf{a}) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{a}\|^2\right) ,$$

Table 2: Various functions of interest for some popular exponential families. For all the cases shown in the table, \mathbf{x} is the sufficient statistic. Note that for the Gaussian examples the variance σ is assumed to be constant. The number of trials, N , for the binomial and multinomial examples is also assumed to be constant.

Distribution	$p(\mathbf{x}; \theta)$	μ	$\phi(\mu)$	$d_\phi(\mathbf{x}, \mu)$
1-D Gaussian	$\frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp(-\frac{(x-a)^2}{2\sigma^2})$	a	$\frac{1}{2\sigma^2} \mu^2$	$\frac{1}{2\sigma^2} (x - \mu)^2$
1-D Poisson	$\frac{\lambda^x \exp(-\lambda)}{x!}$	λ	$\mu \log \mu - \mu$	$x \log(\frac{x}{\mu}) - (x - \mu)$
1-D Bernoulli	$q^x (1 - q)^{1-x}$	q	$\mu \log \mu + (1 - \mu) \log(1 - \mu)$	$x \log(\frac{x}{\mu}) + (1 - x) \log(\frac{1-x}{1-\mu})$
1-D Binomial	$\frac{N!}{(x)!(N-x)!} q^x (1 - q)^{N-x}$	Nq	$\mu \log(\frac{\mu}{N}) + (N - \mu) \log(\frac{N-\mu}{N})$	$x \log(\frac{x}{\mu}) + (N - x) \log(\frac{N-x}{N-\mu})$
1-D Exponential	$\lambda \exp(-\lambda x)$	$1/\lambda$	$-\log \mu - 1$	$\frac{x}{\mu} - \log(\frac{x}{\mu}) - 1$
d -D Sph. Gaussian	$\frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp(-\frac{\ \mathbf{x}-\mathbf{a}\ ^2}{2\sigma^2})$	\mathbf{a}	$\frac{1}{2\sigma^2} \ \mu\ ^2$	$\frac{1}{2\sigma^2} \ \mathbf{x} - \mu\ ^2$
d -D Multinomial	$\frac{N!}{\prod_{j=1}^d x_j!} \prod_{j=1}^d q_j^{x_j}$	$[Nq_j]_{j=1}^{d-1}$	$\sum_{j=1}^d \mu_j \log(\frac{\mu_j}{N})$	$\sum_{j=1}^d x_j \log(\frac{x_j}{\mu_j})$

Distribution	θ	$\psi(\theta)$	dom(ψ)	dom(ϕ)	I_ψ
1-D Gaussian	$\frac{a}{\sigma^2}$	$\frac{\sigma^2}{2} \theta^2$	\mathbb{R}	\mathbb{R}	\mathbb{R}
1-D Poisson	$\log \lambda$	$\exp(\theta)$	\mathbb{R}	\mathbb{R}_+	\mathbb{N}
1-D Bernoulli	$\log(\frac{q}{1-q})$	$\log(1 + \exp(\theta))$	\mathbb{R}	$[0, 1]$	$\{0, 1\}$
1-D Binomial	$\log(\frac{q}{1-q})$	$N \log(1 + \exp(\theta))$	\mathbb{R}	$[0, N]$	$\{0, 1, \dots, N\}$
1-D Exponential	$-\lambda$	$-\log(-\theta)$	\mathbb{R}_{--}	\mathbb{R}_{++}	\mathbb{R}_{++}
d -D Sph. Gaussian	$\frac{\mathbf{a}}{\sigma^2}$	$\frac{\sigma^2}{2} \ \theta\ ^2$	\mathbb{R}^d	\mathbb{R}^d	\mathbb{R}^d
d -D Multinomial	$[\log(\frac{q_j}{q_i})]_{j=1}^{d-1}$	$N \log(1 + \sum_{j=1}^{d-1} \exp(\theta_j))$	\mathbb{R}^{d-1}	$\{\mu \in \mathbb{R}_+^{d-1}, \mu \leq N\}$	$\{\mathbf{x} \in \mathbb{Z}_+^{d-1}, \mathbf{x} \leq N\}$

where $\mathbf{x}, \mathbf{a} \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}$ is a constant. As shown below, $p(\mathbf{x}, \mathbf{a})$ can be expressed in the canonical form for exponential families with natural parameter $\theta = \frac{\mathbf{a}}{\sigma^2}$ and cumulant function $\psi(\theta) = \frac{\sigma^2}{2} \|\theta\|^2$,

$$\begin{aligned}
 p(\mathbf{x}; \mathbf{a}) &= \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{a}\|^2\right) \\
 &= \exp\left(\langle \mathbf{x}, \frac{\mathbf{a}}{\sigma^2} \rangle - \frac{1}{2\sigma^2} \|\mathbf{a}\|^2 - \frac{1}{2\sigma^2} \|\mathbf{x}\|^2\right) \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \\
 &= \exp\left(\langle \mathbf{x}, \theta \rangle - \frac{\sigma^2}{2} \|\theta\|^2\right) \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}\|^2\right) \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \\
 &= \exp(\langle \mathbf{x}, \theta \rangle - \psi(\theta)) p_0(\mathbf{x}),
 \end{aligned}$$

where $p_0(\mathbf{x})$ is independent of θ . By (11), the expectation parameter for this distribution is given by

$$\mu = \nabla \psi(\theta) = \nabla \left(\frac{\sigma^2}{2} \|\theta\|^2 \right) = \sigma^2 \theta = \mathbf{a}.$$

By using (12), the Legendre dual ϕ of ψ is

$$\phi(\mu) = \langle \mu, \theta \rangle - \psi(\theta) = \left\langle \mu, \frac{\mu}{\sigma^2} \right\rangle - \frac{\sigma^2}{2} \|\theta\|^2 = \frac{\|\mu\|^2}{2\sigma^2}.$$

The corresponding Bregman divergence equals

$$\begin{aligned} d_\phi(\mathbf{x}, \boldsymbol{\mu}) &= \phi(\mathbf{x}) - \phi(\boldsymbol{\mu}) - \langle \mathbf{x} - \boldsymbol{\mu}, \nabla\phi(\boldsymbol{\mu}) \rangle = \frac{\|\mathbf{x}\|^2}{2\sigma^2} - \frac{\|\boldsymbol{\mu}\|^2}{2\sigma^2} - \left\langle \mathbf{x} - \boldsymbol{\mu}, \frac{\boldsymbol{\mu}}{\sigma^2} \right\rangle \\ &= \frac{\|\mathbf{x} - \boldsymbol{\mu}\|^2}{2\sigma^2}. \end{aligned}$$

The function $b_\phi(\mathbf{x})$ in Theorem 4 is given by

$$b_\phi(\mathbf{x}) = \exp(\phi(\mathbf{x}))p_0(\mathbf{x}) = \exp\left(\frac{\|\mathbf{x}\|^2}{2\sigma^2} - \frac{\|\boldsymbol{\mu}\|^2}{2\sigma^2}\right) \frac{1}{\sqrt{(2\pi\sigma^2)^d}} = \frac{1}{\sqrt{(2\pi\sigma^2)^d}},$$

and turns out to be a constant. Thus, $p_{(\psi, \theta)}(\mathbf{x}) = \exp(-d_\phi(\mathbf{x}, \boldsymbol{\mu}))b_\phi(\mathbf{x})$.

Example 10 Another exponential family that is widely used is the family of multinomial distributions:

$$p(\mathbf{x}; \mathbf{q}) = \frac{N!}{\prod_{j=1}^d x_j!} \prod_{j=1}^d q_j^{x_j},$$

where $x_j \in \mathbb{Z}_+$ are frequencies of events, $\sum_{j=1}^d x_j = N$ and $q_j \geq 0$ are probabilities of events, $\sum_{j=1}^d q_j = 1$. As shown below, $p(\mathbf{x}; \mathbf{q})$ can be expressed as the density of an exponential distribution in $\mathbf{x} = \{x_j\}_{j=1}^{d-1}$ with natural parameter $\boldsymbol{\theta} = \{\log(\frac{q_j}{q_d})\}_{j=1}^{d-1}$ and cumulant function $\psi(\boldsymbol{\theta}) = -N \log q_d = N \log(1 + \sum_{j=1}^{d-1} e^{\theta_j})$.

$$\begin{aligned} p(\mathbf{x}; \mathbf{q}) &= \frac{N!}{\prod_{j=1}^d x_j!} \prod_{j=1}^d q_j^{x_j} \\ &= \exp\left(\sum_{j=1}^d x_j \log q_j\right) \frac{N!}{\prod_{j=1}^d x_j!} = \exp\left(\sum_{j=1}^{d-1} x_j \log q_j + x_d \log q_d\right) p_0(\mathbf{x}) \\ &= \exp\left(\sum_{j=1}^{d-1} x_j \log q_j + (N - \sum_{j=1}^{d-1} x_j) \log q_d\right) p_0(\mathbf{x}) \\ &= \exp\left(\sum_{j=1}^{d-1} x_j \log\left(\frac{q_j}{q_d}\right) + N \log q_d\right) p_0(\mathbf{x}) \\ &= \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle + N \log q_d) p_0(\mathbf{x}) = \exp\left(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - N \log\left(\sum_{j=1}^d \frac{q_j}{q_d}\right)\right) p_0(\mathbf{x}) \\ &= \exp\left(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - N \log\left(1 + \sum_{j=1}^{d-1} e^{\theta_j}\right)\right) p_0(\mathbf{x}) = \exp(\langle \mathbf{x}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta})) p_0(\mathbf{x}), \end{aligned}$$

where $p_0(\mathbf{x})$ is independent of $\boldsymbol{\theta}$. The expectation parameter $\boldsymbol{\mu}$ is given by

$$\boldsymbol{\mu} = \nabla\psi(\boldsymbol{\theta}) = \nabla\left(N \log\left(1 + \sum_{j=1}^{d-1} e^{\theta_j}\right)\right) = \left[\frac{N e^{\theta_j}}{1 + \sum_{j=1}^{d-1} e^{\theta_j}}\right]_{j=1}^{d-1} = [N q_j]_{j=1}^{d-1}$$

and the Legendre dual ϕ of ψ is

$$\begin{aligned}\phi(\boldsymbol{\mu}) &= \langle \boldsymbol{\mu}, \boldsymbol{\theta} \rangle - \psi(\boldsymbol{\theta}) = \sum_{j=1}^{d-1} Nq_j \log \left(\frac{q_j}{q_d} \right) + N \log q_d \\ &= \sum_{j=1}^d Nq_j \log q_j = N \sum_{j=1}^d \left(\frac{\mu_j}{N} \right) \log \left(\frac{\mu_j}{N} \right),\end{aligned}$$

where $\mu_d = Nq_d$ so that $\sum_{i=1}^d \mu_j = N$. Note that $\phi(\boldsymbol{\mu})$ is a constant multiple of negative entropy for the discrete probability distribution given by $\{\frac{\mu_j}{N}\}_{j=1}^d$. From Example 2, we know that the corresponding Bregman divergence will be a similar multiple of KL-divergence, i.e.,

$$\begin{aligned}d_\phi(\mathbf{x}, \boldsymbol{\mu}) &= \phi(\mathbf{x}) - \phi(\boldsymbol{\mu}) - \langle \mathbf{x} - \boldsymbol{\mu}, \nabla \phi(\boldsymbol{\mu}) \rangle \\ &= N \sum_{j=1}^d \frac{x_j}{N} \log \left(\frac{x_j}{N} \right) - N \sum_{j=1}^d \frac{\mu_j}{N} \log \left(\frac{\mu_j}{N} \right) - \sum_{j=1}^d (x_j - \mu_j) \left(1 + \log \left(\frac{\mu_j}{N} \right) \right) \\ &= N \sum_{j=1}^d \frac{x_j}{N} \log \left(\frac{x_j/N}{\mu_j/N} \right).\end{aligned}$$

The function $b_\phi(\mathbf{x})$ for this case is given by

$$b_\phi(\mathbf{x}) = \exp(\phi(\mathbf{x}))p_0(\mathbf{x}) = \exp \left(\sum_{j=1}^d x_j \log \left(\frac{x_j}{N} \right) \right) \frac{N!}{\prod_{j=1}^d x_j!} = \frac{\prod_{j=1}^d x_j^{x_j}}{N^N} \frac{N!}{\prod_{j=1}^d x_j!},$$

and $p_{(\psi, \boldsymbol{\theta})}(\mathbf{x}) = \exp(-d_\phi(\mathbf{x}, \boldsymbol{\mu}))b_\phi(\mathbf{x})$.

5. Bregman Soft Clustering

Using the correspondence between regular exponential families and regular Bregman divergences, we now pose the Bregman soft clustering problem as a parameter estimation problem for mixture models based on regular exponential family distributions. We revisit the expectation maximization (EM) framework for estimating mixture densities and develop a Bregman soft clustering algorithm (Algorithm 3) for regular Bregman divergences. We also present the Bregman soft clustering algorithm for a set of data points with non-uniform non-negative weights (or measure). Finally, we show how the hard clustering algorithm can be interpreted as a special case of the soft clustering algorithm and also discuss an alternative formulation of hard clustering in terms of a dual divergence derived from the conjugate function.

5.1 Soft Clustering as Mixture Density Estimation

Given a set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ drawn independently from a stochastic source, consider the problem of modeling the source using a single parametric exponential family distribution. This is the problem of maximum likelihood estimation, or, equivalently, minimum negative log-likelihood estimation of the parameter(s) of a given exponential family distribution. From Theorem 4, minimizing the negative log-likelihood is the same as minimizing the corresponding expected Bregman divergence. Using Proposition 1, we conclude that the optimal distribution is the one with $\boldsymbol{\mu} = \mathbf{E}[X]$ as the expectation parameter, where X is a random variable that takes values in \mathcal{X} following (by the

independence assumption) the empirical distribution over \mathcal{X} . Further, note that the minimum negative log-likelihood of \mathcal{X} under a particular exponential model with log-partition function ψ is the Bregman information of X , i.e., $I_\phi(X)$, up to additive constants, where ϕ is the Legendre conjugate of ψ .

Now, consider the problem of modeling the stochastic source with a mixture of k densities of the same exponential family. The model yields a soft clustering where clusters correspond to the components of the mixture model, and the soft membership of a data point in each cluster is proportional to the probability of the data point being generated by the corresponding density function. For regular Bregman divergences, we define the *Bregman soft clustering problem* as that of learning the maximum likelihood parameters $\Gamma = \{\theta_h, \pi_h\}_{h=1}^k \equiv \{\mu_h, \pi_h\}_{h=1}^k$ of a mixture model of the form

$$p(\mathbf{x}|\Gamma) = \sum_{h=1}^k \pi_h p_{(\psi, \theta_h)}(\mathbf{x}) = \sum_{h=1}^k \pi_h \exp(-d_\phi(\mathbf{x}, \mu_h)) b_\phi(\mathbf{x}), \quad (16)$$

where the last equality follows from Theorem 4. Since the mixture components are all assumed to be from the same family, the above problem is a special case of the general maximum likelihood parameter estimation problem for mixture models and can be solved by applying the EM algorithm.

5.2 EM for Mixture Models Based on Bregman Divergences

Algorithm 2 describes the well known application of EM for mixture density estimation. This algorithm has the property that the likelihood of the data, $L_{\mathcal{X}}(\Gamma)$ is non-decreasing at each iteration. Further, if there exists at least one local maximum for the likelihood function, then the algorithm will converge to a local maximum of the likelihood. For more details, the reader is referred to Collins (1997); McLachlan and Krishnan (1996) and Bilmes (1997).

The Bregman soft clustering problem is to estimate the maximum likelihood parameters for the mixture model given in (16). Using the Bregman divergence viewpoint, we get a simplified version of the above EM algorithm that we call the Bregman soft clustering algorithm (Algorithm 3). Using Proposition 1, the computationally intensive M-step turns out to be straightforward to solve. In fact, the Bregman divergence viewpoint gives an alternative interpretation of a well known efficient EM scheme applicable to learning a mixture of exponential distributions (Redner and Walker, 1984). The resulting update equations are similar to those for learning mixture models of identity covariance Gaussians. Note that these equations are applicable to mixtures of any regular exponential distributions, as long as \mathbf{x} is the (minimal) sufficient statistic vector.

It is important to note that the simplification of the M-step is applicable only when the parameterization is with respect to the expectation parameter space, i.e., when d_ϕ corresponding to an exponential family is known. Otherwise, if the parameterization is with respect to the natural parameter space, i.e., the functional form for a family is known in terms of its cumulant ψ and natural parameters θ , the problem

$$\phi(\mathbf{x}) = \sup_{\theta \in \mathbb{R}^d} (\langle \theta, \mathbf{x} \rangle - \psi(\theta)), \quad (17)$$

needs to be solved to obtain $\phi(\mathbf{x})$. Since the function to be maximized in (17) is precisely the log-likelihood of the exponential family density (with respect to an appropriate measure), the transformation is equivalent to solving a maximum likelihood estimation problem (with a single sample), which is computationally expensive for several exponential family distributions. In such a situation, transforming the problem to the expectation space need not lead to any tangible computational bene-

Algorithm 2 Standard EM for Mixture Density Estimation

Input: Set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subseteq \mathbb{R}^d$, number of clusters k .

Output: Γ^\dagger : local maximizer of $L_{\mathcal{X}}(\Gamma) = \prod_{i=1}^n (\sum_{h=1}^k \pi_h p_{\psi, \theta_h}(\mathbf{x}_i))$ where $\Gamma = \{\theta_h, \pi_h\}_{h=1}^k$, soft partitioning $\{\{p(h|\mathbf{x}_i)\}_{h=1}^k\}_{i=1}^n$.

Method:

Initialize $\{\theta_h, \pi_h\}_{h=1}^k$ with some $\theta_h \in \Theta$, and $\pi_h \geq 0$, $\sum_{h=1}^k \pi_h = 1$

repeat

{The Expectation Step (E-step)}

for $i = 1$ to n **do**

for $h = 1$ to k **do**

$$p(h|\mathbf{x}_i) \leftarrow \frac{\pi_h p_{(\psi, \theta_h)}(\mathbf{x}_i)}{\sum_{h'=1}^k \pi_{h'} p_{(\psi, \theta_{h'})}(\mathbf{x}_i)}$$

end for

end for

{The Maximization Step (M-step)}

for $h = 1$ to k **do**

$$\pi_h \leftarrow \frac{1}{n} \sum_{i=1}^n p(h|\mathbf{x}_i)$$

$$\theta_h \leftarrow \operatorname{argmax}_{\theta} \sum_{i=1}^n \log(p_{(\psi, \theta)}(\mathbf{x}_i) p(h|\mathbf{x}_i))$$

end for

until convergence

return $\Gamma^\dagger = \{\theta_h, \pi_h\}_{h=1}^k$

Algorithm 3 Bregman Soft Clustering

Input: Set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathcal{S} \subseteq \mathbb{R}^d$, Bregman divergence $d_\phi : \mathcal{S} \times \operatorname{ri}(\mathcal{S}) \mapsto \mathbb{R}$, number of clusters k .

Output: Γ^\dagger , local maximizer of $\prod_{i=1}^n (\sum_{h=1}^k \pi_h b_\phi(\mathbf{x}_i) \exp(-d_\phi(\mathbf{x}_i, \mu_h)))$ where $\Gamma = \{\mu_h, \pi_h\}_{h=1}^k$, soft partitioning $\{\{p(h|\mathbf{x}_i)\}_{h=1}^k\}_{i=1}^n$

Method:

Initialize $\{\mu_h, \pi_h\}_{h=1}^k$ with some $\mu_h \in \operatorname{ri}(\mathcal{S})$, $\pi_h \geq 0$, and $\sum_{h=1}^k \pi_h = 1$

repeat

{The Expectation Step (E-step)}

for $i = 1$ to n **do**

for $h = 1$ to k **do**

$$p(h|\mathbf{x}_i) \leftarrow \frac{\pi_h \exp(-d_\phi(\mathbf{x}_i, \mu_h))}{\sum_{h'=1}^k \pi_{h'} \exp(-d_\phi(\mathbf{x}_i, \mu_{h'}))}$$

end for

end for

{The Maximization Step (M-step)}

for $h = 1$ to k **do**

$$\pi_h \leftarrow \frac{1}{n} \sum_{i=1}^n p(h|\mathbf{x}_i)$$

$$\mu_h \leftarrow \frac{\sum_{i=1}^n p(h|\mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n p(h|\mathbf{x}_i)}$$

end for

until convergence

return $\Gamma^\dagger = \{\mu_h, \pi_h\}_{h=1}^k$

fits. However, if the Bregman divergence d_ϕ corresponding to an exponential family is either known or easy to compute from the natural parameterization, then Algorithm 3 is computationally much more efficient. In fact, in some situations it may be easier to design regular Bregman divergences for mixture modeling of data than to come up with an appropriate exponential family. Such situations can take full advantage of the computationally efficient Bregman soft clustering algorithm.

The following result shows how Proposition 1 and Theorem 4 can be used to simplify the M-step of Algorithm 2. Using this result, we then show that Algorithms 2 and 3 are exactly equivalent for regular Bregman divergences and exponential families. Note that Proposition 4 has appeared in various forms in the literature (see, for example, Redner and Walker (1984); McLachlan and Krishnan (1996)). We give an alternative proof using Bregman divergences.

Proposition 4 *For a mixture model with density given by (16), the maximization step for the density parameters in the EM algorithm (Algorithm 2), $\forall h, 1 \leq h \leq k$, reduces to:*

$$\boldsymbol{\mu}_h = \frac{\sum_{i=1}^n p(h|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^n p(h|\mathbf{x}_i)}. \quad (18)$$

Proof The maximization step for the density parameters in the EM algorithm, $\forall h, 1 \leq h \leq k$, is given by

$$\boldsymbol{\theta}_h = \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^n \log(p_{(\psi, \boldsymbol{\theta})}(\mathbf{x}_i))p(h|\mathbf{x}_i).$$

For the given mixture density, the component densities, $\forall h, 1 \leq h \leq k$, are given by

$$p_{(\psi, \boldsymbol{\theta}_h)}(\mathbf{x}) = b_\phi(\mathbf{x}) \exp(-d_\phi(\mathbf{x}, \boldsymbol{\mu}_h)).$$

Substituting the above into the maximization step, we obtain the update equations for the expectation parameters $\boldsymbol{\mu}_h$, $1 \leq h \leq k$,

$$\begin{aligned} \boldsymbol{\mu}_h &= \operatorname{argmax}_{\boldsymbol{\mu}} \sum_{i=1}^n \log(b_\phi(\mathbf{x}_i) \exp(-d_\phi(\mathbf{x}_i, \boldsymbol{\mu})))p(h|\mathbf{x}_i) \\ &= \operatorname{argmax}_{\boldsymbol{\mu}} \sum_{i=1}^n (\log(b_\phi(\mathbf{x}_i)) - d_\phi(\mathbf{x}_i, \boldsymbol{\mu}))p(h|\mathbf{x}_i) \\ &= \operatorname{argmin}_{\boldsymbol{\mu}} \sum_{i=1}^n d_\phi(\mathbf{x}_i, \boldsymbol{\mu})p(h|\mathbf{x}_i) \text{ (as } b_\phi(\mathbf{x}) \text{ is independent of } \boldsymbol{\mu}) \\ &= \operatorname{argmin}_{\boldsymbol{\mu}} \sum_{i=1}^n d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) \frac{p(h|\mathbf{x}_i)}{\sum_{i'=1}^n p(h|\mathbf{x}_{i'})}. \end{aligned}$$

From Proposition 1, we know that the expected Bregman divergence is minimized by the expectation of \mathbf{x} , i.e.,

$$\operatorname{argmin}_{\boldsymbol{\mu}} \sum_{i=1}^n d_\phi(\mathbf{x}_i, \boldsymbol{\mu}) \frac{p(h|\mathbf{x}_i)}{\sum_{i'=1}^n p(h|\mathbf{x}_{i'})} = \frac{\sum_{i=1}^n p(h|\mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n p(h|\mathbf{x}_i)}.$$

Therefore, the update equation for the parameters is just a weighted averaging step,

$$\boldsymbol{\mu}_h = \frac{\sum_{i=1}^n p(h|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^n p(h|\mathbf{x}_i)}, \quad \forall h, 1 \leq h \leq k.$$

■

The update equations for the posterior probabilities (E-step) $\forall \mathbf{x} \in \mathcal{X}, \forall h, 1 \leq h \leq k$, are given by

$$p(h|\mathbf{x}) = \frac{\pi_h \exp(-d_\phi(\mathbf{x}, \boldsymbol{\mu}_h))}{\sum_{h'=1}^k \pi_{h'} \exp(-d_\phi(\mathbf{x}, \boldsymbol{\mu}_{h'}))}$$

as the $b_\phi(\mathbf{x})$ factor cancels out. The prior update equations are independent of the parametric form of the densities and remain unaltered. Hence, for a mixture model with density given by (16), the EM algorithm (Algorithm 2) reduces to the Bregman soft clustering algorithm (Algorithm 3).

So far we have considered the Bregman soft clustering problem for a set \mathcal{X} where all the elements are equally important and assumed to have been independently sampled from some particular exponential distribution. In practice, it might be desirable to associate weights v_i with the individual samples such that $\sum_i v_i = 1$ and optimize a weighted log-likelihood function. A slight modification to the M-step of the Bregman soft clustering algorithm is sufficient to address this new optimization problem. The E-step remains identical and the new update equations for the M-step, $\forall h, 1 \leq h \leq k$, are given by

$$\begin{aligned} \pi_h &= \sum_{i=1}^n v_i p(h|\mathbf{x}_i), \\ \boldsymbol{\mu}_h &= \frac{\sum_{i=1}^n v_i p(h|\mathbf{x}_i) \mathbf{x}_i}{\sum_{i=1}^n v_i p(h|\mathbf{x}_i)}. \end{aligned}$$

Finally, we note that the Bregman hard clustering algorithm is a limiting case of the above soft clustering algorithm. For every convex function ϕ and positive constant β , $\beta\phi$ is also a convex function with the corresponding Bregman divergence $d_{\beta\phi} = \beta d_\phi$. In the limit, when $\beta \rightarrow \infty$, the posterior probabilities in the E-step take values in $\{0, 1\}$ and hence, the E and M steps of the soft clustering algorithm reduce to the assignment and re-estimation steps of the hard clustering algorithm.

5.3 An Alternative Formulation for Bregman Clustering

In earlier sections, the Bregman divergence was measured with the data points as the first argument and the cluster representative as the second argument. Since Bregman divergences are not symmetric (with the exception of squared Euclidean distance), we now consider an alternative formulation of Bregman clustering where cluster representatives are the first argument of the Bregman divergence. Using Legendre duality, we show that this alternate formulation is equivalent to our original Bregman clustering problem in a dual space using a different, but uniquely determined Bregman divergence.

We focus on the hard clustering case. Let X be a random variable that takes values in $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ following a positive probability measure ν . Then the alternative Bregman hard clustering problem is to find clusters $\{\mathcal{X}_h\}_{h=1}^k$ and corresponding representatives $\{\boldsymbol{\mu}_h\}_{h=1}^k$ that solve

$$\min_{\{\boldsymbol{\mu}_h\}_{h=1}^k} \sum_{h=1}^k \sum_{\mathbf{x}_i \in \mathcal{X}_h} v_i d_\phi(\boldsymbol{\mu}_h, \mathbf{x}_i). \tag{19}$$

As mentioned earlier, Bregman divergences are convex in the first argument and hence, the resulting optimization problem for each cluster is convex so there is a unique optimal representative for each

cluster. However, unlike in the original formulation, the optimal cluster representative is not always the expectation and depends on the Bregman divergence d_ϕ .

It is interesting to note that this alternative formulation, though seemingly different, reduces to the original formulation with an appropriate representation. Let ϕ be the generating convex function of d_ϕ such that $(\text{int}(\text{dom}(\phi)), \phi)$ is a convex function of Legendre type and let $(\text{int}(\text{dom}(\psi)), \psi)$ be the corresponding Legendre dual. Then for any $\mathbf{x}, \mathbf{y} \in \text{int}(\text{dom}(\phi))$, the Bregman divergence $d_\phi(\mathbf{x}, \mathbf{y}) = d_\psi(\boldsymbol{\theta}_\mathbf{y}, \boldsymbol{\theta}_\mathbf{x})$ where d_ψ is the Bregman divergence derived from ψ and $\boldsymbol{\theta}_\mathbf{x} = \nabla\phi(\mathbf{x}), \boldsymbol{\theta}_\mathbf{y} = \nabla\phi(\mathbf{y})$ (Appendix A, Property 6). Using the above property, we can restate the alternative Bregman clustering problem in the dual space. More specifically, let $\mathcal{X}^\theta = \{\boldsymbol{\theta}_{\mathbf{x}_i}\}_{i=1}^n$ where $\boldsymbol{\theta}_{\mathbf{x}_i} = \nabla\phi(\mathbf{x}_i), \forall \mathbf{x}_i, 1 \leq i \leq n$, and let $\boldsymbol{\theta}_h = \nabla\phi(\boldsymbol{\mu}_h), \forall \boldsymbol{\mu}_h, 1 \leq h \leq k$. Then the hard clustering problem (19) can be expressed as

$$\min_{\{\boldsymbol{\theta}_h\}_{h=1}^k} \sum_{h=1}^k \sum_{\boldsymbol{\theta}_{\mathbf{x}_i} \in \mathcal{X}_h^\theta} v_i d_\psi(\boldsymbol{\theta}_{\mathbf{x}_i}, \boldsymbol{\theta}_h). \tag{20}$$

where \mathcal{X}_h^θ correspond to cluster h in the dual space. It is now straightforward to see that this is our original Bregman hard clustering problem for the set \mathcal{X}^θ consisting of the dual data points with the same measure v and the dual Bregman divergence d_ψ . The optimal cluster representative in this dual space is given by the expectation, which is easy to compute. The efficiency of this approach is based on the same premise as the efficient EM scheme for exponential families, i.e, the M-step can be simplified if there is an easy transition to the dual space.

6. Lossy Compression and Generalized Loss Functions

In this section, we study the connection between Bregman clustering algorithms and lossy compression schemes. In particular, we focus on the relationship of our work with Shannon’s rate distortion theory, showing connections between learning mixtures of exponential distributions, the Bregman soft clustering problem and the rate distortion problem where distortion is measured using a regular Bregman divergence (Banerjee et al., 2004a). Then we show that all these problems involve a trade-off between compression and loss in Bregman information. The information bottleneck method (Tishby et al., 1999) emerges as a special case of this viewpoint. We restrict our attention to regular exponential families and regular Bregman divergences in this section.

6.1 Rate Distortion Theory for Bregman Divergences

Rate distortion theory (Berger, 1971; Berger and Gibson, 1998) deals with the fundamental limits of quantizing a stochastic source $X \sim p(x), x \in \mathcal{X}$, using a random variable \hat{X} over a reproduction alphabet $\hat{\mathcal{X}}$ typically assumed to embed the source alphabet \mathcal{X} , i.e., $\mathcal{X} \subseteq \hat{\mathcal{X}}$. In the rate distortion setting, the performance of a quantization scheme is determined in terms of the rate, i.e., the average number of bits for encoding a symbol, and the expected distortion between the source and the reproduction random variables based on an appropriate distortion function $d : \mathcal{X} \times \hat{\mathcal{X}} \mapsto \mathbb{R}_+$. The central problem in rate distortion theory (Cover and Thomas, 1991) is to compute the rate distortion function $R(D)$, which is defined as the minimum achievable rate for a specified level of expected distortion D , and can be mathematically expressed as

$$R(D) = \min_{p(\hat{x}|x): E_{X, \hat{X}}[d(X, \hat{X})] \leq D} I(X; \hat{X}), \tag{21}$$

where $I(X; \hat{X})$ is the mutual information of X and \hat{X} .

The rate distortion problem is a convex problem that involves optimizing over the probabilistic assignments $p(\hat{x}|x)$ and can be theoretically solved using the Blahut-Arimoto algorithm (Arimoto, 1972; Blahut, 1972; Csiszár, 1974; Cover and Thomas, 1991). However, numerical computation of the rate distortion function through the Blahut-Arimoto algorithm is often infeasible in practice, primarily due to lack of knowledge of the optimal support of the reproduction random variable. An efficient solution for addressing this problem is the mapping approach (Banerjee et al., 2004a; Rose, 1994), where one solves a related problem that assumes cardinality k for the support of the reproduction random variable. In this setting, the optimization is over the assignments as well as the support set, i.e.,

$$\min_{\substack{\hat{\mathcal{X}}_s, p(\hat{x}|x) \\ |\hat{\mathcal{X}}_s|=k}} I(X; \hat{X}) + \beta_D E_{X, \hat{X}}[d(X, \hat{X})] , \quad (22)$$

where β_D is the optimal Lagrange multiplier that depends on the chosen tolerance level D of the expected distortion and $\hat{\mathcal{X}}_s$ is the optimal support of the reproduction random variable with cardinality k . We shall refer to the above problem (22) as the rate distortion problem with a support set of finite cardinality (RDFC). It can be shown (Berger, 1971) that the RDFC problem and the original rate distortion problem have identical solutions when the cardinality of the optimal support set is less than or equal to k , which is known to be true for cases without an analytical solution (Banerjee et al., 2004a).

Our analysis connects the Bregman soft clustering problem to the RDFC problem following results from Banerjee et al. (2004a), which extend previous work (Rose, 1994; Gray and Neuhoff, 1998) that related kmeans clustering to vector quantization and rate-distortion based on squared Euclidean distortion. Let Z, \hat{Z} denote suitable sufficient statistic representations of X, \hat{X} so that the distortion can be measured by a Bregman divergence d_ϕ in the sufficient statistic space. The RDFC problem can now be stated directly in terms of Z and \hat{Z} as

$$\min_{\substack{\hat{\mathcal{Z}}_s, p(\hat{z}|z) \\ |\hat{\mathcal{Z}}_s|=k}} I(Z; \hat{Z}) + \beta_D E_{Z, \hat{Z}}[d_\phi(Z, \hat{Z})] , \quad (23)$$

where $\hat{\mathcal{Z}}_s$ is the optimal support of the reproduction random variable with cardinality k .

Unlike the basic rate distortion problem (21), the RDFC problem (23) is no longer a convex problem since it involves optimization over both $\hat{\mathcal{Z}}_s$ and $p(\hat{z}|z)$. However, when either of the arguments is fixed, the resulting sub-problem can be solved exactly. In particular, when $\hat{\mathcal{Z}}_s$ is known, then the RDFC problem reduces to that of optimizing over $p(\hat{z}|z)$, which is a feasible convex problem and can be exactly solved by the Blahut-Arimoto algorithm (Csiszár, 1974). Similarly, when the assignments $p(\hat{z}|z)$ are known, the RDFC problem only involves minimizing the expected distortion measured in terms of a Bregman divergence and can be exactly solved using Proposition 1. Thus the objective function in (23) can be greedily minimized by alternately optimizing over the individual arguments, yielding a solution that is locally optimal. The details of this analysis and resulting algorithm can be found in Banerjee et al. (2004a).

Interestingly, it can be shown (Banerjee et al., 2004a) that the RDFC problem based on a regular Bregman divergence is exactly equivalent to the the maximum likelihood mixture estimation problem based on a uniquely determined exponential family when the source distribution in the rate distortion setting equals the empirical distribution over the sampled data points.

Theorem 7 (Banerjee et al. (2004a)) *Consider a source $Z \sim p(z)$, where $p(z)$ is the empirical distribution over the samples. Then the RDFC problem (23) for the source Z with regular Bregman divergence d_ϕ , variational parameter β_D , and reproduction random variable \hat{Z} with $|\hat{Z}| = k$ is equivalent to the maximum likelihood mixture estimation problem based on the regular exponential family $\mathcal{F}_{\beta_D \psi}$ with number of mixture components set to k (ψ is the conjugate of ϕ).*

From Section 5, we know that the maximum likelihood mixture estimation problem for any regular exponential family is equivalent to the Bregman soft clustering problem for the corresponding regular Bregman divergence. Using this in conjunction with Theorem 7, we obtain the following equivalence relation between the RDFC problem and the Bregman soft clustering problem.

Theorem 8 *Consider a source $Z \sim p(z)$, where $p(z)$ is the empirical distribution over the samples. Then the RDFC problem (23) for the source Z with regular Bregman divergence d_ϕ , variational parameter β_D , and reproduction random variable \hat{Z} with $|\hat{Z}| = k$ is equivalent to the Bregman soft clustering problem (16) based on the Bregman divergence $d_{\beta_D \phi}$ with number of clusters set to k .*

From the above theorem, it follows that Algorithm 3 can be used to solve the RDFC problem. Note that the update steps for $p(h|\mathbf{x})$ and π_h in Algorithm 3 exactly correspond to the updates of $p(\hat{z}|z)$ and $p(\hat{z})$ in the Blahut-Arimoto step in Algorithm 1 of Banerjee et al. (2004a) for solving the RDFC problem. The update of μ_h in Algorithm 3 is equivalent to the update of \hat{z} in the support estimation step in Algorithm 1 of Banerjee et al. (2004a). From the viewpoint of alternate minimization, the order of the three updates $p(\hat{z}|z)$, $p(\hat{z})$ and \hat{z} is interchangeable and does not affect the local optimality guarantees, although different orderings may lead to different solutions.

The Bregman soft clustering problem corresponds to the RDFC problem and not to the basic rate distortion problem (21). However, as mentioned earlier, both the problems yield the same solution for the rate distortion function when the optimal support set $|\hat{Z}_s|$ is finite and k is sufficiently large. The solution is the rate distortion function and refers to the asymptotic rate (Cover and Thomas, 1991) that can be achieved for a given distortion, when we are allowed to code the source symbols in blocks of size m with $m \rightarrow \infty$.

It is also possible to consider a related rate distortion problem where the source symbols are coded using blocks of size 1. The resultant rate distortion function is referred to as the “scalar” or “order 1” rate distortion function $R_1(D)$ (Gray and Neuhoff, 1998). The problem is solved by performing hard assignments of the source symbols to the closest codebook members, which is similar to the assignment step in the Bregman hard clustering problem. In fact, the “order 1” or “1-shot” rate distortion problem, assuming a known finite cardinality of the optimal reproduction support set, turns out to be exactly equivalent to the Bregman hard clustering problem.

6.2 Compression vs. Bregman Information Trade-off

We now provide yet another view of the RDFC problem (and hence, Bregman soft clustering) as a lossy compression problem where the objective is to balance the trade-off between compression and preservation of Bregman information. Intuitively, the reproduction random variable \hat{Z} is a coarser representation of the source random variable Z with less “information” than Z . In rate distortion theory, the loss in “information” is quantified by the expected Bregman distortion between Z and \hat{Z} . The following theorem, which is along the same lines as Theorem 1, provides a direct way of quantifying the intuitive loss in “information” in terms of Bregman information.

Theorem 9 (Banerjee et al. (2004a)) *The expected Bregman distortion between the source and the reproduction random variables is exactly equal to the loss in Bregman information due to compression, i.e.,*

$$E_{Z,\hat{Z}}[d_\phi(Z,\hat{Z})] = I_\phi(Z) - I_\phi(\hat{Z}),$$

where $\hat{Z} = E_{Z|\hat{Z}}[Z]$.

The RDFC problem (23) can, therefore, be viewed as an optimization problem involving a trade-off between the mutual information $I(Z;\hat{Z})$ that measures the compression, and the loss in Bregman information $I_\phi(Z) - I_\phi(\hat{Z})$. Since the source random variable Z is known, the Bregman information $I_\phi(Z)$ is fixed and minimizing the expected distortion is equivalent to maximizing the Bregman information of the compressed random variable \hat{Z} . Hence, this constrained form of the RDFC problem (23) can be written as:

$$\min_{p(\hat{z}|z)} \{I(Z;\hat{Z}) - \beta I_\phi(\hat{Z})\}, \quad (24)$$

where β is the variational parameter corresponding to the desired point in the rate distortion curve and $\hat{Z} = E_{Z|\hat{Z}}[Z]$. The variational parameter β determines the trade-off between the achieved compression and the preserved Bregman information.

6.2.1 INFORMATION BOTTLENECK REVISITED

We now demonstrate how the information bottleneck (IB) method of Tishby et al. (1999) can be derived from the RDFC problem (24) for a suitable choice of Bregman divergence.

Let $Y \sim p(y)$, $y \in \mathcal{Y}$ be a random variable. Let the sufficient statistic random vector Z corresponding to a source X be the conditional distribution of Y given X , i.e., $Z = p(Y|X)$. Z is just a concrete representation of the possibly abstract source X . Similarly, the random variable $\hat{Z} = p(Y|\hat{X})$ represents the reproduction random variable \hat{X} . This choice of sufficient statistic mapping is appropriate when the joint distribution of the random variables X and Y contains all the relevant information about X . For the above choice of sufficient statistic mapping, an additional constraint that \hat{Z} is the conditional expectation of Z leads to the lossy compression problem (24) where we need to find the optimal assignments that balance the trade-off between compression and the loss in Bregman information. Now, from Example 6 in Section 3.1, the Bregman information $I_\phi(\hat{Z})$ of the random variable \hat{Z} that takes values over the set of conditional distributions $\{p(Y|\hat{x})\}$ with probability $p(\hat{x})$ is the same as the mutual information $I(\hat{X};Y)$ of \hat{X} and Y (when the Bregman divergence is the KL-divergence). Hence, the problem (24) reduces to

$$\min_{p(\hat{x}|x)} \{I(X;\hat{X}) - \beta I(\hat{X};Y)\}, \quad (25)$$

since $p(\hat{x}|x) = p(\hat{z}|z)$ and $I(X;\hat{X}) = I(Z;\hat{Z})$, where β is the variational parameter. This is identical to the IB formulation (Tishby et al., 1999). Our framework reveals that the IB assumption that the mutual information with respect to another random variable Y holds all the relevant information for comparing the different source entities is equivalent to assuming that (a) $p(Y|X)$ is the appropriate sufficient statistic representation, and (b) the KL-divergence between the conditional distributions of Y is the appropriate distortion measure. Further, the assumption about the conditional independence of Y and \hat{X} given X , i.e., the Markov chain condition $Y \leftrightarrow X \leftrightarrow \hat{X}$, is equivalent to the constraint that \hat{Z} is the conditional expectation of Z , i.e., $\hat{Z} = p(Y|\hat{X}) = E_{X|\hat{X}}[p(Y|X)] = E_{Z|\hat{Z}}[Z]$.

Thus the information bottleneck problem is seen to be a special case of the RDFC problem (23), and hence also of the Bregman soft clustering problem and mixture estimation problem for exponential families. In particular, IB is exactly equivalent to the mixture estimation problem based on the exponential family corresponding to KL-divergence, i.e., the multinomial family (Collins et al., 2001). Further, the iterative IB algorithm is the same as the EM algorithm for multinomial distributions (Slonim and Weiss, 2002), and also the Bregman soft clustering algorithm using KL-divergence.

7. Experiments

There are a number of experimental results in existing literature (MacQueen, 1967; Linde et al., 1980; Buzo et al., 1980; Dhillon et al., 2003; Nigam et al., 2000) that illustrate the usefulness of specific Bregman divergences and the corresponding Bregman clustering algorithms in important application domains. The classical kmeans algorithm, which is a special case of the Bregman hard clustering algorithm for the squared Euclidean distance has been successfully applied to a large number of domains where a Gaussian distribution assumption is valid. Besides this, there are at least two other domains where special cases of Bregman clustering methods have been shown to provide good results.

The first is the text-clustering domain where the information-theoretic clustering algorithm (Dhillon et al., 2003) and the EM algorithm on a mixture of multinomials based on the naive Bayes assumption (Nigam et al., 2000) have been applied. These algorithms are, respectively, special cases of the Bregman hard and soft clustering algorithms for KL-divergence, and have been shown to provide high quality results on large real datasets such as the 20-Newsgroups, Reuters and Dmoz text datasets. This success is not unexpected as text documents can be effectively modeled using multinomial distributions where the corresponding Bregman divergence is just the KL-divergence between word distributions.

Speech coding is another domain where a special case of the Bregman clustering algorithm based on the Itakura-Saito distance, namely the Linde-Buzo-Gray (LBG) algorithm (Linde et al., 1980; Buzo et al., 1980), has been successfully applied. Speech power spectra tend to follow exponential family densities of the form $p(x) = \lambda e^{-\lambda x}$ whose corresponding Bregman divergence is the Itakura-Saito distance (see Table 2).

Since special cases of Bregman clustering algorithms have already been shown to be effective in various domains, we do not experimentally re-evaluate the Bregman clustering algorithms against other methods. Instead, we only focus on showing that the quality of the clustering depends on the appropriateness of the Bregman divergence. In particular we study Bregman clustering of data generated from mixture of exponential family distributions using the corresponding Bregman divergence as well as non-matching divergences. The results indicate that the cluster quality is best when the Bregman divergence corresponding to the generative model is employed.

We performed two experiments using datasets of increasing level of difficulty. For our first experiment, we created three 1-dimensional datasets of 100 samples each, based on mixture models of Gaussian, Poisson and Binomial distributions respectively. All the mixture models had three components with equal priors centered at 10, 20 and 40 respectively. The standard deviation σ of the Gaussian densities was set to 5 and the number of trials N of the Binomial distribution was set to 100 so as to make the three models somewhat similar to each other, in the sense that the variance is approximately the same for all the models. Figure 1 shows the density functions of the generative

models. The datasets were then each clustered using three versions of the Bregman hard clustering

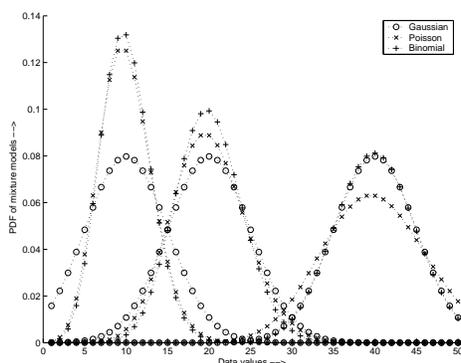


Figure 1: Generative models for data sets used in experiment 1

Table 3: Clustering results for the first data set. Columns 2-4 correspond to the normalized mutual information between original and predicted clusters obtained by applying the Bregman clustering algorithm corresponding to the Bregman divergences $d_{Gaussian}$, $d_{Poisson}$ and $d_{Binomial}$ respectively

Generative Model	$d_{Gaussian}$	$d_{Poisson}$	$d_{Binomial}$
Gaussian	0.701 ± 0.033	0.633 ± 0.043	0.641 ± 0.035
Poisson	0.689 ± 0.063	0.734 ± 0.057	0.694 ± 0.059
Binomial	0.769 ± 0.061	0.746 ± 0.048	0.825 ± 0.046

algorithm corresponding to the Bregman divergences obtained from the Gaussian (kmeans), Poisson and Binomial distributions respectively. The quality of the clustering was measured in terms of the normalized mutual information¹¹ (Strehl and Ghosh, 2002) between the predicted clusters and original clusters (based on the actual generating mixture component), and the results were averaged over 10 trials. Table 3 shows the normalized mutual information values for the different divergences and datasets. From Table 3, we can see that clustering quality is significantly better when the Bregman divergence used in the clustering algorithm matches that of the generative model.

The second experiment involved a similar kind of comparison of clustering algorithms for multi-dimensional datasets drawn from multivariate Gaussian, Binomial and Poisson distributions respectively. The datasets were sampled from mixture models with 15 overlapping components and had 2000 10-dimensional samples each. The results of the Bregman clustering algorithms shown in Table 4 lead to the same conclusion as before, i.e., the choice of the Bregman divergence used for clustering is important for obtaining good quality clusters.

In practice, an important issue that needs to be addressed is: what is the appropriate Bregman divergence for a given application? In certain situations, it may be possible to realistically characterize the data generative process using a mixture of exponential family distributions. In such a scenario, especially in the absence of a better methodology, using the divergence corresponding to

11. It is meaningless to compare the clustering objective function values as they are different for the three versions of the Bregman clustering algorithm.

Table 4: Clustering results for the second set of data sets

Generative Model	d_{Gaussian}	d_{Poisson}	d_{Binomial}
Gaussian	0.728 ± 0.005	0.661 ± 0.007	0.669 ± 0.005
Poisson	0.792 ± 0.013	0.815 ± 0.014	0.802 ± 0.013
Binomial	0.823 ± 0.006	0.833 ± 0.011	0.849 ± 0.012

the exponential family seems appropriate. In general, however, the divergence used for clustering need not necessarily have to be the one corresponding to the generative model. The final choice should depend on the relevant application, i.e., the divergence should capture the similarity properties desirable in the application, and need not necessarily depend on how the data was actually generated.

8. Related Work

This work is largely inspired by three broad and overlapping ideas. First, an information theoretic viewpoint of the clustering problem is invaluable. Such considerations occur in several techniques, from classical vector quantization (Gersho and Gray, 1992) to information theoretic clustering (Dhillon et al., 2003) and the information bottleneck method (Tishby et al., 1999). In particular, the information theoretic hard clustering (Dhillon et al., 2003) approach solved the problem of distributional clustering with a formulation involving loss in Shannon’s mutual information. In this paper, we have significantly generalized that work by proposing techniques for obtaining optimal quantizations by minimizing loss in Bregman information corresponding to arbitrary Bregman divergences.

Second, our soft clustering approach is based on the relationship between Bregman divergences and exponential family distributions and the suitability of Bregman divergences as distortion or loss functions for data drawn from exponential distributions. It has been previously shown (Amari and Nagaoka, 2001; Azoury and Warmuth, 2001) that the KL-divergence, which is the most natural distance measure for comparing two members $p_{(\psi, \theta)}$ and $p_{(\psi, \tilde{\theta})}$ of an exponential family, is always a Bregman divergence. In particular, it is the Bregman divergence $d_{\psi}(\theta, \tilde{\theta})$ corresponding to the cumulant function ψ of the exponential family. In our work, we extend this concept to say that the Bregman divergence of the Legendre conjugate of the cumulant function is a natural distance function for the data drawn according to a mixture model based on that exponential family.

The third broad idea is that many learning algorithms can be viewed as solutions for minimizing loss functions based on Bregman divergences (Censor and Zenios, 1998). Elegant techniques for the design of algorithms and the analysis of relative loss bounds in the online learning setting extensively use this framework (Azoury and Warmuth, 2001). In the unsupervised learning setting, use of this framework typically involves development of alternate minimization procedures (Csiszár and Tusnády, 1984). For example, Pietra et al. (2001); Wang and Schuurmans (2003) analyze and develop iterative alternate projection procedures for solving unsupervised optimization problems that involve objective functions based on Bregman divergences under various kinds of constraints. Further, Collins et al. (2001) develop a generalization of PCA for exponential families using loss functions based on the corresponding Bregman divergences and propose alternate minimization schemes for solving the problem.

On a larger context, there has been research in various fields that has focussed on generalized notions of distances and on extending known methodologies to the general setting (Rao, 1982). Grünwald and Dawid (2004) recently extended the ‘redundancy-capacity theorem’ of information theory to arbitrary discrepancy measures. As an extension of Shannon’s entropy (Cover and Thomas, 1991), they introduced generalized entropy measures that are (not necessarily differentiable) concave functions of probability distributions. Just as Shannon’s entropy is the minimum number of bits (on an average) required to encode a stochastic source, the generalized entropy measures correspond to the infimum of a general class of loss functions in a game theoretic setting. Restricting their results to our setting, the generalized entropy is equivalent to the concave function $-\phi$, where ϕ determines the Bregman divergence d_ϕ . However, our framework is applicable to arbitrary vectors (or functions), whereas Grünwald and Dawid (2004) focus only on probability distributions.

As we discussed in Section 6, our treatment of clustering is very closely tied to rate distortion theory (Berger, 1971; Berger and Gibson, 1998; Gray and Neuhoff, 1998). The results presented in the paper extend vector quantization methods (Gersho and Gray, 1992) to a large class of distortion measures. Further, building on the work of Rose (1994), our results provide practical ways of computing the rate-distortion function when distortion is measured by a Bregman divergence. In addition, the results also establish a connection between the rate distortion problem with Bregman divergences and the mixture model estimation problem for exponential families (Banerjee et al., 2004a).

In the literature, there are clustering algorithms that involve minimizing loss functions based on distortion measures that are somewhat different from Bregman divergences. For example, Modha and Spangler (2003) present the *convex-kmeans* clustering algorithm for distortion measures that are always non-negative and convex in the second argument, using the notion of a generalized centroid. Bregman divergences, on the other hand, are not necessarily convex in the second argument. Linde et al. (1980) consider distortion measures of the form $d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T A(\mathbf{x})(\mathbf{x} - \mathbf{y})$ where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and $A(\mathbf{x})$ is a $d \times d$ positive definite matrix, as loss functions for vector quantization. Although such distortions are Bregman divergences in some cases, e.g., when $A(\mathbf{x})$ is a constant matrix, in general one has to solve a convex optimization problem to compute the optimal representative when using the above $d(\mathbf{x}, \mathbf{y})$.

9. Concluding Remarks

In this paper, we have presented hard and soft clustering algorithms to minimize loss functions involving Bregman divergences. Our analysis presents a unified view of an entire class of centroid based parametric clustering algorithms. First, in the hard-clustering framework, we show that a *kmeans* type iterative relocation scheme solves the Bregman hard-clustering problem for all Bregman divergences. Further, using a related result, we see that Bregman divergences are the only distortion functions for which such a centroid-based clustering scheme is possible. Second, we formally show that there is a one-to-one correspondence between regular exponential families and regular Bregman divergences. This result is useful in developing an alternative interpretation of the EM algorithm for learning mixtures of exponential distributions, eventually resulting in a class of Bregman soft-clustering algorithms. Our formulation also turns out to be closely tied to the rate distortion theory for Bregman divergences.

As discussed in the paper, special cases of our analysis have been discovered and widely used by researchers in applications ranging from speech coding to text clustering. There are three salient features of this framework that make these results particularly useful for real-life applications. First, the computational complexity of each iteration of the entire class of Bregman clustering algorithms is linear in the number of data-points. Hence the algorithms are scalable and appropriate for large-scale machine learning tasks. Second, the modularity of the proposed class of algorithms is evident from the fact that only one component in the proposed schemes, i.e., the Bregman divergence used in the assignment step, needs to be changed to obtain an algorithm for a new loss function. This simplifies the implementation and application of this class of algorithms to various data types. Third, the algorithms discussed are also applicable to mixed data types that are commonly encountered in real applications. Since a convex combination of convex functions is always convex, one can have different convex functions appropriately chosen for different subsets of features. The Bregman divergence corresponding to a convex combination of the component functions can now be used to cluster the data, thus vastly increasing the scope of the proposed techniques.

Acknowledgments

We would like to thank Peter Grünwald and an anonymous referee for their constructive comments that greatly improved the presentation of the paper. We thank Manfred Warmuth for pointing out that a rigorous proof of (13) was required, and Alex Smola and S. V. N. Vishwanathan for their valuable comments on Section 4. Part of this research was supported by an IBM PhD fellowship, NSF grants IIS-0307792, IIS-0325116, CCF-0431257, NSF CAREER Award No. ACI-0093404 and Texas Advanced Research Program grant 003658-0431-2001.

Appendix A. Properties of Bregman Divergences

In this section, we list some well-known useful properties of Bregman divergences.

Properties of Bregman Divergences

Let $\phi : \mathcal{S} \mapsto \mathbb{R}$ be a strictly convex, differentiable function defined on a convex set $\mathcal{S} = \text{dom}(\phi) \subseteq \mathbb{R}^d$ and let $d_\phi : \mathcal{S} \times \text{ri}(\mathcal{S}) \mapsto [0, \infty)$ be its Bregman divergence, i.e., $d_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \mathbf{x} - \mathbf{y}, \nabla\phi(\mathbf{y}) \rangle$. Then, the following properties are true.

1. **Non-negativity.** $d_\phi(\mathbf{x}, \mathbf{y}) \geq 0$, $\forall \mathbf{x} \in \mathcal{S}, \mathbf{y} \in \text{ri}(\mathcal{S})$, and equality holds if and only if $\mathbf{x} = \mathbf{y}$.
2. **Convexity.** d_ϕ is always convex in the first argument, but not necessarily convex in the second argument. Squared Euclidean distance and KL-divergence are examples of Bregman divergences that are convex in both their arguments, but the Bregman divergence corresponding to the strictly convex function $\phi(x) = x^3$, defined on \mathbb{R}_+ , given by $d_\phi(x, y) = x^3 - y^3 - 3(x - y)y^2$ an example divergence that is not convex in y .
3. **Linearity.** Bregman divergence is a linear operator i.e., $\forall \mathbf{x} \in \mathcal{S}, \mathbf{y} \in \text{ri}(\mathcal{S})$,

$$\begin{aligned} d_{\phi_1 + \phi_2}(\mathbf{x}, \mathbf{y}) &= d_{\phi_1}(\mathbf{x}, \mathbf{y}) + d_{\phi_2}(\mathbf{x}, \mathbf{y}), \\ d_{c\phi}(\mathbf{x}, \mathbf{y}) &= cd_\phi(\mathbf{x}, \mathbf{y}) \quad (\text{for } c \geq 0). \end{aligned}$$

4. **Equivalence classes.** The Bregman divergences of functions that differ only in affine terms are identical i.e., if $\phi(\mathbf{x}) = \phi_0(\mathbf{x}) + \langle \mathbf{b}, \mathbf{x} \rangle + c$ where $\mathbf{b} \in \mathbb{R}^d$ and $c \in \mathbb{R}$, then $d_\phi(\mathbf{x}, \mathbf{y}) = d_{\phi_0}(\mathbf{x}, \mathbf{y}), \forall \mathbf{x} \in \mathcal{S}, \mathbf{y} \in \text{ri}(\mathcal{S})$. Hence, the set of all strictly convex, differentiable functions on a convex set \mathcal{S} can be partitioned into equivalence classes of the form

$$[\phi_0] = \{\phi \mid d_\phi(\mathbf{x}, \mathbf{y}) = d_{\phi_0}(\mathbf{x}, \mathbf{y}) \forall \mathbf{x} \in \mathcal{S}, \mathbf{y} \in \text{ri}(\mathcal{S})\}.$$

5. **Linear separation.** The locus of all the points $\mathbf{x} \in \mathcal{S}$ that are equidistant from two fixed points $\mu_1, \mu_2 \in \text{ri}(\mathcal{S})$ in terms of a Bregman divergence is a hyperplane, i.e., the partitions induced by Bregman divergences have linear separators given by

$$\begin{aligned} d_\phi(\mathbf{x}, \mu_1) &= d_\phi(\mathbf{x}, \mu_2) \\ \Rightarrow \phi(\mathbf{x}) - \phi(\mu_1) - \langle \mathbf{x} - \mu_1, \nabla \phi(\mu_1) \rangle &= \phi(\mathbf{x}) - \phi(\mu_2) - \langle \mathbf{x} - \mu_2, \nabla \phi(\mu_2) \rangle \\ \Rightarrow \langle \mathbf{x}, \nabla \phi(\mu_2) - \nabla \phi(\mu_1) \rangle &= (\phi(\mu_1) - \phi(\mu_2)) - (\langle \mu_1, \nabla \phi(\mu_1) \rangle - \langle \mu_2, \nabla \phi(\mu_2) \rangle) \end{aligned}$$

6. **Dual Divergences.** Bregman divergences obtained from a Legendre function ϕ and its conjugate ψ satisfy the duality property:

$$d_\phi(\mu_1, \mu_2) = \phi(\mu_1) + \psi(\theta_2) - \langle \mu_1, \theta_2 \rangle = d_\psi(\theta_2, \theta_1),$$

where $\mu_1, \mu_2 \in \text{ri}(\mathcal{S})$ are related to $\theta_1, \theta_2 \in \text{ri}(\Theta)$ by the Legendre transformation.

7. **Relation to KL-divergence.** Let \mathcal{F}_ψ be an exponential family with ψ as the cumulant function. Then the KL divergence between two members $p_{(\psi, \theta_1)}$ and $p_{(\psi, \theta_2)}$ in \mathcal{F}_ψ corresponding to natural parameters θ_1 and θ_2 can be expressed as a Bregman divergence in two possible ways. In particular,

$$KL(p_{(\psi, \theta_1)} \parallel p_{(\psi, \theta_2)}) = d_\phi(\mu_1, \mu_2) = d_\psi(\theta_2, \theta_1)$$

where μ_1 and μ_2 are the expectation parameters corresponding to θ_1 and θ_2 . Further, if $\psi(\mathbf{0}) = 0$, then $p_{(\psi, \mathbf{0})}(\mathbf{x}) = p_0(\mathbf{x})$ is itself a valid probability density and $KL(p_{(\psi, \theta)} \parallel p_{(\psi, \mathbf{0})}) = \phi(\mu)$, where $\mu = \nabla \psi(\theta)$.

8. **Generalized Pythagoras theorem.** For any $\mathbf{x}_1 \in \mathcal{S}$ and $\mathbf{x}_2, \mathbf{x}_3 \in \text{ri}(\mathcal{S})$, the following three-point property holds:

$$d_\phi(\mathbf{x}_1, \mathbf{x}_3) = d_\phi(\mathbf{x}_1, \mathbf{x}_2) + d_\phi(\mathbf{x}_2, \mathbf{x}_3) - \langle \mathbf{x}_1 - \mathbf{x}_2, \nabla \phi(\mathbf{x}_3) - \nabla \phi(\mathbf{x}_2) \rangle. \quad (26)$$

When $\mathbf{x}_1, \mathbf{x}_2$ and \mathbf{x}_3 are such that $\mathbf{x}_1 \in \mathcal{S}'$ where \mathcal{S}' is a convex subset of \mathcal{S} and \mathbf{x}_2 is given by

$$\mathbf{x}_2 = \underset{\mathbf{x} \in \mathcal{S}'}{\text{argmin}} d_\phi(\mathbf{x}, \mathbf{x}_3),$$

then the inner product term in (26) becomes negative and we have,

$$d_\phi(\mathbf{x}_1, \mathbf{x}_2) + d_\phi(\mathbf{x}_2, \mathbf{x}_3) \leq d_\phi(\mathbf{x}_1, \mathbf{x}_3).$$

When the convex subset \mathcal{S}' is an affine set, then the inner product term is zero giving rise to an equality.

Necessary and Sufficient Conditions for a Bregman Divergence

A divergence measure $d : \mathcal{S} \times \text{ri}(\mathcal{S}) \mapsto [0, \infty)$ is a Bregman divergence if and only if there exists $\mathbf{a} \in \text{ri}(\mathcal{S})$ such that the function $\phi_{\mathbf{a}}(\mathbf{x}) = d(\mathbf{x}, \mathbf{a})$ satisfies the following conditions:

1. $\phi_{\mathbf{a}}$ is strictly convex on \mathcal{S} and differentiable on $\text{ri}(\mathcal{S})$.
2. $d(\mathbf{x}, \mathbf{y}) = d_{\phi_{\mathbf{a}}}(\mathbf{x}, \mathbf{y})$, $\forall \mathbf{x} \in \mathcal{S}, \mathbf{y} \in \text{ri}(\mathcal{S})$ where $d_{\phi_{\mathbf{a}}}$ is the Bregman divergence associated with $\phi_{\mathbf{a}}$.

It is easy to see the sufficiency property from the second condition. To prove that the conditions are necessary as well, we note that for any strictly convex, differentiable function ϕ , the Bregman divergence evaluated with a fixed value for the second argument differs from it only by a linear term, i.e.,

$$\begin{aligned} \phi_{\mathbf{a}}(\mathbf{x}) = d_{\phi}(\mathbf{x}, \mathbf{a}) &= \phi(\mathbf{x}) - \phi(\mathbf{a}) - \langle \mathbf{x} - \mathbf{a}, \nabla\phi(\mathbf{a}) \rangle \\ &= \phi(\mathbf{x}) + \langle \mathbf{b}, \mathbf{x} \rangle + c, \end{aligned}$$

where $\mathbf{b} = -\nabla\phi(\mathbf{a})$ and $c = \langle \mathbf{a}, \nabla\phi(\mathbf{a}) \rangle - \phi(\mathbf{a})$. Hence, $\phi_{\mathbf{a}}$ is also strictly convex and differentiable and the Bregman divergences associated with ϕ and $\phi_{\mathbf{a}}$ are identical.

Appendix B. Proof of Exhaustiveness Result

This appendix is based on results reported in Banerjee et al. (2005) and is included in this paper for the sake of completeness. The results discussed here show the exhaustiveness of Bregman divergences with respect to the property proved in Proposition 1.

Theorem 10 (Banerjee et al. (2005)) *Let $F : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_+$ be a continuous and differentiable function $F(x, y)$ with continuous partial derivatives $\frac{\partial F}{\partial x}$ and $\frac{\partial F}{\partial y}$ such that $F(x, x) = 0, \forall x \in \mathbb{R}$. For all sets $\mathcal{X} \subseteq \mathbb{R}$ and all probability measures \mathbf{v} over \mathcal{X} , if the random variable X takes values in \mathcal{X} following \mathbf{v} such that $y^* = E_{\mathbf{v}}[X]$ is the unique minimizer of $E_{\mathbf{v}}[F(X, y)]$ over all $y \in \mathbb{R}$, i.e., if*

$$\underset{y \in \mathbb{R}}{\operatorname{argmin}} E_{\mathbf{v}}[F(X, y)] = E_{\mathbf{v}}[X] \tag{27}$$

then $F(x, y)$ is a Bregman divergence, i.e., $F(x, y) = d_{\phi}(x, y)$ for some strictly convex, differentiable function $\phi : \mathbb{R} \mapsto \mathbb{R}$.

Proof Since the optimality property in (27) is true for all \mathcal{X} and \mathbf{v} , we give a constructive argument with a particular choice of \mathcal{X} and \mathbf{v} . Let $\mathcal{X} = \{a, b\} \subset \mathbb{R}$ where $a \neq b$, and let \mathbf{v} be $\{p, q\}$, with $p, q \in (0, 1)$ and $p + q = 1$ so that $E_{\mathbf{v}}[X] = pa + qb$. Then from (27),

$$pF(a, y) + qF(b, y) = E_{\mathbf{v}}[F(X, y)] \geq E_{\mathbf{v}}[F(X, E_{\mathbf{v}}[X])] = pF(a, pa + qb) + qF(b, pa + qb)$$

$\forall y \in \mathbb{R}$. If we consider the left-hand-side as a function of y , it equals the right-hand-side at $y = y^* \doteq E_{\mathbf{v}}[X] = pa + qb$. Therefore, we must have

$$p \frac{\partial F(a, y^*)}{\partial y} + q \frac{\partial F(b, y^*)}{\partial y} = 0. \tag{28}$$

Substituting $p = (y^* - b)/(a - b)$ and rearranging terms yields

$$\frac{1}{(y^* - a)} \frac{\partial F(a, y^*)}{\partial y} = \frac{1}{(y^* - b)} \frac{\partial F(b, y^*)}{\partial s}.$$

Since a, b and p are arbitrary, the above equality implies that the function

$$\frac{1}{(y - x)} \frac{\partial F(x, y)}{\partial y}$$

is independent of x . Thus we can write, for some function H ,

$$\frac{\partial F(x, y)}{\partial y} = (y - x)H(y), \tag{29}$$

for some continuous function H .

Now define function ϕ by

$$\phi(y) = \int_0^y \int_0^{y'} H(t) dt dy'.$$

Then ϕ is differentiable with $\phi(0) = \phi'(0) = 0$, $\phi''(y) = H(y)$. Integration by parts for (29) leads to

$$F(x, y) - F(x, x) = \int_x^y (y' - x)H(y') dy' = \phi(x) - \phi(y) - \phi'(y)(x - y).$$

Since $F(x, x) = 0$, the non-negativity of F implies that ϕ is a convex function.

It remains to show that ϕ is strictly convex. Suppose ϕ is not strictly convex. Then there exists an interval $I = [\ell_1, \ell_2]$ such that $\ell_1 < \ell_2$ and $\phi'(y) = (\phi(\ell_1) - \phi(\ell_2))/(\ell_1 - \ell_2)$ for all $y \in I$. Consider the set $\mathcal{X} = \{\ell_1, \ell_2\}$ with $\nu = \{\frac{1}{2}, \frac{1}{2}\}$. It is easy to check that any $y \in I$ is a minimizer of $E_\nu[F(X, y)]$. This is a contradiction, and so ϕ must be strictly convex. ■

It is possible to get rid of the condition that $\frac{\partial F}{\partial y}$ has to be continuous by proper mollification arguments (Banerjee et al., 2005). Further, it is possible to generalize the result to functions in more than one dimension, i.e., $F : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_+$.

Theorem 11 (Banerjee et al. (2005)) *Let $F : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_+$ be a continuous function such that $F(\mathbf{x}, \mathbf{x}) = 0, \forall \mathbf{x} \in \mathbb{R}^d$, and the second order partial derivatives $\frac{\partial^2 F}{\partial \mathbf{x}_i \partial \mathbf{x}_j}, 1 \leq i, j, \leq d$, are all continuous. For all sets $\mathcal{X} \subseteq \mathbb{R}^d$ and all probability measures ν over \mathcal{X} , if the random variable X takes values in \mathcal{X} following ν such that $\mathbf{y} = E_\nu[X]$ is the unique minimizer of $E_\nu[F(X, \mathbf{y})]$ over all $\mathbf{y} \in \mathbb{R}^d$, i.e., if*

$$\operatorname{argmin}_{\mathbf{y} \in \mathbb{R}^d} E_\nu[F(X, \mathbf{y})] = E_\nu[X],$$

then $F(\mathbf{x}, \mathbf{y})$ is a Bregman divergence, i.e., $F(\mathbf{x}, \mathbf{y}) = d_\phi(\mathbf{x}, \mathbf{y})$ for some strictly convex, differentiable function $\phi : \mathbb{R}^d \mapsto \mathbb{R}$.

The proof of Theorem 11 builds on the intuition of the proof of Theorem 10, but is more involved and hence skipped; the interested reader is referred to Banerjee et al. (2005).

Appendix C. Proof of Theorem 3

This appendix provides a proof of Theorem 3 in Section 4.3 and related results. Most of the ideas used in our analysis are from Section 9.1 of Barndorff-Nielsen (1978). For the sake of completeness, we give detailed proofs of the results. We begin with definitions. Let P_0 be any non-negative bounded measure on \mathbb{R}^d and $\mathcal{F}_\psi = \{p_{(\psi, \theta)}, \theta \in \Theta \subseteq \mathbb{R}^d\}$ be a regular exponential family with cumulant function ψ and base measure P_0 , as discussed in Section 4.1. Without loss of generality, let P_0 be a probability measure.¹² Let I_ψ be the support of P_0 (Definition 6) and hence, of all the probability distributions in \mathcal{F}_ψ . Let ϕ be the conjugate of ψ so that $(\text{int}(\text{dom}(\phi)), \phi)$ and (Θ, ψ) are Legendre duals of each other.

Lemma 3 For any $\theta \in \Theta$ and $\mathbf{x} \in \mathbb{R}^d$,

$$\langle \theta, \mathbf{x} \rangle - \psi(\theta) \leq -\log \left(\inf_{\mathbf{u} \in \mathbb{R}^d, \|\mathbf{u}\|_2=1} P_0[\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x} \rangle] \right) \quad (30)$$

where $X \sim P_0$. Hence

$$\inf_{\mathbf{u} \in \mathbb{R}^d, \|\mathbf{u}\|_2=1} P_0[\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x} \rangle] > 0 \text{ implies that } \mathbf{x} \in \text{dom}(\phi).$$

Proof Let \mathbf{u}_θ be the unit vector in the direction of θ . Given any $\mathbf{x} \in \mathbb{R}^d$, it is possible to divide \mathbb{R}^d into two half spaces $\mathcal{G}_1 = \{\mathbf{x}' \in \mathbb{R}^d \mid \langle \mathbf{u}_\theta, \mathbf{x}' \rangle < \langle \mathbf{u}_\theta, \mathbf{x} \rangle\}$ and $\mathcal{G}_2 = \{\mathbf{x}' \in \mathbb{R}^d \mid \langle \mathbf{u}_\theta, \mathbf{x}' \rangle \geq \langle \mathbf{u}_\theta, \mathbf{x} \rangle\}$. For any θ , we have

$$\begin{aligned} 1 &= \int_{\mathbf{x}' \in \mathbb{R}^d} \exp(\langle \theta, \mathbf{x}' \rangle - \psi(\theta)) dP_0(\mathbf{x}') \\ \Rightarrow \exp(\psi(\theta)) &= \int_{\mathbf{x}' \in \mathbb{R}^d} \exp(\langle \theta, \mathbf{x}' \rangle) dP_0(\mathbf{x}'). \end{aligned}$$

Partitioning the integral over \mathbb{R}^d into \mathcal{G}_1 and \mathcal{G}_2 , we obtain

$$\begin{aligned} \exp(\psi(\theta)) &= \int_{\mathbf{x}' \in \mathcal{G}_1} \exp(\langle \theta, \mathbf{x}' \rangle) dP_0(\mathbf{x}') + \int_{\mathbf{x}' \in \mathcal{G}_2} \exp(\langle \theta, \mathbf{x}' \rangle) dP_0(\mathbf{x}') \\ &\geq \int_{\mathbf{x}' \in \mathcal{G}_2} \exp(\langle \theta, \mathbf{x}' \rangle) dP_0(\mathbf{x}') \\ &\geq \exp(\langle \theta, \mathbf{x} \rangle) \int_{\mathbf{x}' \in \mathcal{G}_2} dP_0(\mathbf{x}') \quad (\text{since } \langle \mathbf{u}_\theta, \mathbf{x}' \rangle \geq \langle \mathbf{u}_\theta, \mathbf{x} \rangle \text{ for } \mathbf{x}' \in \mathcal{G}_2) \\ &= \exp(\langle \theta, \mathbf{x} \rangle) P_0[\langle \mathbf{u}_\theta, X \rangle \geq \langle \mathbf{u}_\theta, \mathbf{x} \rangle] \\ &\geq \exp(\langle \theta, \mathbf{x} \rangle) \inf_{\mathbf{u}} P_0[\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x} \rangle]. \end{aligned}$$

On taking logarithms and re-arranging terms, we obtain (30).

From (30), $\inf_{\mathbf{u}} P_0[\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x} \rangle] > 0$ implies that $\forall \theta, \langle \theta, \mathbf{x} \rangle - \psi(\theta) < \infty$, so that

$$\phi(\mathbf{x}) = \sup_{\theta} (\langle \theta, \mathbf{x} \rangle - \psi(\theta)) < \infty,$$

12. Since any non-negative bounded measure can be simply converted to a probability measure by a multiplicative constant, our analysis remains practically unchanged in the general case, except for an additive constant to the cumulant function.

i.e., $\mathbf{x} \in \text{dom}(\phi)$. ■

We now prove the claim of Theorem 3 that $I_\Psi \subseteq \text{dom}(\phi)$.

Proof of Theorem 3 Let $\mathbf{x}_0 \in I_\Psi$ and let \mathbf{u} be any unit vector. Let $H(\mathbf{u}, \mathbf{x}_0)$ be the hyperplane through \mathbf{x}_0 with unit normal \mathbf{u} . Let $\mathcal{H}(\mathbf{u}, \mathbf{x}_0)$ be the closed half-space determined by the hyperplane $H(\mathbf{u}, \mathbf{x}_0)$, i.e., $\mathcal{H}(\mathbf{u}, \mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^d \mid \langle \mathbf{u}, \mathbf{x} \rangle \geq \langle \mathbf{u}, \mathbf{x}_0 \rangle\}$. Using this notation, we give separate proofs for the cases when P_0 is absolutely continuous with respect to the counting measure and with respect to the Lebesgue measure.

Let P_0 be absolutely continuous with respect to the counting measure. By definition, $\mathbf{x}_0 \in \mathcal{H}(\mathbf{u}, \mathbf{x}_0)$. Since $\mathbf{x}_0 \in I_\Psi$, applying Definition 6 to the set $I = \{\mathbf{x}_0\}$ we have $p_{(\Psi, \theta)}(\mathbf{x}_0) > 0$. Hence $p_0(\mathbf{x}_0) > 0$ as the exponential family distribution is absolutely continuous with respect to P_0 . Therefore, the closed half-space $\mathcal{H}(\mathbf{u}, \mathbf{x}_0)$ has a positive measure of at least $p_0(\mathbf{x}_0)$ for any unit vector \mathbf{u} , i.e.,

$$\begin{aligned} P_0[\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x}_0 \rangle] &\geq p_0(\mathbf{x}_0) > 0 \quad \forall \mathbf{u} \\ \text{so that} \quad \inf_{\mathbf{u}} P_0[\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x}_0 \rangle] &\geq p_0(\mathbf{x}_0) > 0. \end{aligned}$$

From Lemma 3, it follows that $\mathbf{x}_0 \in \text{dom}(\phi)$. Therefore, $I_\Psi \subseteq \text{dom}(\phi)$.

Now we consider the case when P_0 is absolutely continuous with respect to the Lebesgue measure. If $\mathbf{x}_0 \in I_\Psi$, then $\forall I \subseteq \mathbb{R}^d$ with $\mathbf{x}_0 \in I$ and $\int_I d\mathbf{x} > 0$, we have

$$\int_I dP_0(\mathbf{x}) > 0.$$

Note that since $\mathbf{x}_0 \in \mathcal{H}(\mathbf{u}, \mathbf{x}_0)$ and $\int_{\mathcal{H}(\mathbf{u}, \mathbf{x}_0)} d\mathbf{x} > 0$, we must have

$$\int_{\mathcal{H}(\mathbf{u}, \mathbf{x}_0)} dP_0(\mathbf{x}) > 0 \quad \forall \mathbf{u}.$$

Hence, $P_0(\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x}_0 \rangle) > 0, \forall \mathbf{u}$. Since the set of unit vectors is a compact set, $\inf_{\mathbf{u}} P_0(\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x}_0 \rangle)$ is achieved at some unit vector \mathbf{u}^* , so that

$$\inf_{\mathbf{u}} P_0(\langle \mathbf{u}, X \rangle \geq \langle \mathbf{u}, \mathbf{x}_0 \rangle) = P_0(\langle \mathbf{u}^*, X \rangle \geq \langle \mathbf{u}^*, \mathbf{x}_0 \rangle) > 0.$$

Again, Lemma 3 implies that $\mathbf{x}_0 \in \text{dom}(\phi)$ so that $I_\Psi \subseteq \text{dom}(\phi)$. ■

Finally, we present a related result from Barndorff-Nielsen (1978) involving the closed convex hull of I_Ψ and $\text{dom}(\phi)$. The result is not essential to the paper, but is relevant, and interesting in its own right.

Theorem 12 (Barndorff-Nielsen (1978)) *Let I_Ψ be as in Definition 6. Let C_Ψ be the closure of the convex hull of I_Ψ , i.e., $C_\Psi = \text{co}(I_\Psi)$. Then,*

$$\text{int}(C_\Psi) \subseteq \text{dom}(\phi) \subseteq C_\Psi$$

where ϕ is the conjugate of Ψ .

Note that Theorem 12 does not imply Theorem 3.

References

- N. I. Akhizer. *The Classical Moment Problem and some related questions in analysis*. Hafner Publishing Company, 1965.
- S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995.
- S. Amari and H. Nagaoka. *Methods of Information Geometry*. American Mathematical Society, 2001.
- S. Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18:14–20, 1972.
- K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.
- A. Banerjee, I. Dhillon, J. Ghosh, and S. Merugu. An information theoretic analysis of maximum likelihood mixture estimation for exponential families. In *Proc. 21st International Conference on Machine Learning (ICML)*, 2004a.
- A. Banerjee, X. Guo, and H. Wang. Optimal Bregman prediction and Jensen’s equality. In *Proc. International Symposium on Information Theory (ISIT)*, 2004b.
- A. Banerjee, X. Guo, and H. Wang. On the optimality of conditional expectation as a Bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669, July 2005.
- O. Barndorff-Nielsen. *Information and Exponential Families in Statistical Theory*. Wiley Publishers, 1978.
- C. Berg, J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Springer-Verlag, 1984.
- T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice-Hall, 1971.
- T. Berger and J. D. Gibson. Lossy source coding. *IEEE Transactions on Information Theory*, 44(6):2691–2723, 1998.
- J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden markov models. Technical Report ICSI-TR-97-02, University of Berkeley, 1997.
- R. E. Blahut. Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, 18:460–473, 1972.
- L. M. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

- A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(5):562–574, 1980.
- Y. Censor and S. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1998.
- M. Collins. The EM algorithm. Technical report, Department of Computer and Information Science, University of Pennsylvania, 1997.
- M. Collins, S. Dasgupta, and R. Schapire. A generalization of principal component analysis to the exponential family. In *Proc. of the 14th Annual Conference on Neural Information Processing Systems (NIPS)*, 2001.
- T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.
- I. Csiszár. On the computation of rate distortion functions. *IEEE Transactions of Information Theory*, IT-20:122:124, 1974.
- I. Csiszár. Why least squares and maximum entropy? An axiomatic approach to inference for linear inverse problems. *The Annals of Statistics*, 19(4):2032–2066, 1991.
- I. Csiszár. Generalized projections for non-negative functions. *Acta Mathematica Hungarica*, 68(1-2):161–185, 1995.
- I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics and Decisions, Supplement Issue*, 1(1):205–237, 1984.
- A. Devinatz. The representation of functions as Laplace-Stieltjes integrals. *Duke Mathematical Journal*, 24:481–498, 1955.
- I. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3(4):1265–1287, 2003.
- R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.
- W. Ehm, M. G. Genton, and T. Gneiting. Stationary covariances associated with exponentially convex functions. *Bernoulli*, 9(4):607–615, 2003.
- J. Forster and M. K. Warmuth. Relative expected instantaneous loss bounds. In *Proc. of the 13th Annual Conference on Computational Learning Theory (COLT)*, pages 90–99, 2000.
- A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.
- R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998.
- P. D. Grünwald and A. Dawid. Game theory, maximum entropy, minimum discrepancy, and robust bayesian decision theory. *Annals of Statistics*, 32(4), 2004.

- P. D. Grünwald and P. Vitányi. Kolmogorov complexity and information theory with an interpretation in terms of questions and answers. *Journal of Logic, Language and Information*, 12(4): 497–529, 2003.
- A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.
- D. Kazakos and P.P. Kazakos. Spectral distance measures between Gaussian processes. *IEEE Transactions on Automatic Control*, 25(5):950–959, 1980.
- Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.
- J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
- G. J. McLachlan and T. Krishnan. *The EM algorithm and Extensions*. Wiley-Interscience, 1996.
- D. Modha and S. Spangler. Feature weighting in k-means clustering. *Machine Learning*, 52(3): 217–237, 2003.
- K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.
- M. Palus. On entropy rates of dynamical systems and Gaussian processes. *Physics Letters A*, 227 (5-6):301–308, 1997.
- S. D. Pietra, V. D. Pietra, and J. Lafferty. Duality and auxiliary functions for Bregman distances. Technical Report CMU-CS-01-109, School of Computer Science, Carnegie Mellon University, 2001.
- C. R. Rao. Diversity and dissimilarity coefficients: A unified approach. *Journal of Theoretical Population Biology*, 21:24–43, 1982.
- R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.
- R. T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics. Princeton University Press, 1970.
- K. Rose. A mapping approach to rate-distortion computation and analysis. *IEEE Transactions on Information Theory*, 40(6):1939–1952, 1994.
- N. Slonim and Y. Weiss. Maximum likelihood and the information bottleneck. In *Proc. 16th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 335–342, 2002.
- A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. *Journal of Machine Learning Research*, 3(3):583–617, 2002.
- N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of the 37th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

- M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report TR 649, Dept. of Statistics, University of California at Berkeley, 2003.
- S. Wang and D. Schuurmans. Learning continuous latent variable models with Bregman divergences. In *Proc. IEEE International Conference on Algorithmic Learning Theory*, 2003.

Combining Information Extraction Systems Using Voting and Stacked Generalization

Georgios Sigletos

SIGLETOS@IIT.DEMOKRITOS.GR

Georgios Paliouras

PALIOURG@IIT.DEMOKRITOS.GR

Constantine D. Spyropoulos

COSTASS@IIT.DEMOKRITOS.GR

*Institute of Informatics and Telecommunications
National Centre for Scientific Research (NCSR) "Demokritos"
Aghia Paraskeyh, 153 10, Athens, Greece*

Michalis Hatzopoulos

MIKE@DI.UOA.GR

*Department of Informatics and Telecommunications
University of Athens
Panepistimiopolis, 157 71, Athens, Greece*

Editor: William Cohen

Abstract

This article investigates the effectiveness of voting and stacked generalization -also known as stacking- in the context of information extraction (IE). A new stacking framework is proposed that accommodates well-known approaches for IE. The key idea is to perform cross-validation on the base-level data set, which consists of text documents annotated with relevant information, in order to create a meta-level data set that consists of feature vectors. A classifier is then trained using the new vectors. Therefore, base-level IE systems are combined with a common classifier at the meta-level. Various voting schemes are presented for comparing against stacking in various IE domains. Well known IE systems are employed at the base-level, together with a variety of classifiers at the meta-level. Results show that both voting and stacking work better when relying on probabilistic estimates by the base-level systems. Voting proved to be effective in most domains in the experiments. Stacking, on the other hand, proved to be consistently effective over all domains, doing comparably or better than voting and always better than the best base-level systems. Particular emphasis is also given to explaining the results obtained by voting and stacking at the meta-level, with respect to the varying degree of similarity in the output of the base-level systems.

Keywords: stacking, voting, information extraction, cross-validation

1 Introduction

One of the most interesting topics in supervised machine learning is learning how to combine the individual predictions of multiple classifiers. The motivation derives from the opportunity of obtaining higher prediction accuracy at meta-level, while treating classifiers as *black boxes*, i.e., using only their output, without considering the details of their implementation. *Stacked generalization* or *stacking* (Wolpert, 1992) is a common scheme that deals with the task of learning a meta-level classifier to combine the predictions of multiple base-level classifiers. The success of stacking arises from its ability to exploit the diversity in the predictions of base-level classifiers and thus predicting with higher accuracy at meta-level. In contrast, no learning takes

place when *voting* on the predictions of multiple classifiers. Voting is typically used as a baseline against which the performance of stacking is compared.

Research on voting and stacking has primarily focused on classification. Each training instance in the domain of interest is represented by a vector $\langle x_1 \dots x_n, y \rangle$, where $x_1 \dots x_n$ is a set of attribute values or features, and y is the class value describing a particular event in the domain, which is to be recognized at runtime. In order to classify a new vector $\langle x_1 \dots x_n \rangle$, the predictions of the base-level classifiers form a new feature vector, which is assigned the class value y either by the meta-level classifier or by voting. Cross-validation in the base-level set of feature vectors is required by stacking, in order to create the entire set of meta-level vectors by the predictions of the base-level classifiers, and thus train the meta-level classifier.

In this article we investigate the effectiveness of voting and stacking on the task of Information Extraction (IE). IE is a form of shallow text processing that involves the population of a predefined template with relevant fragments extracted from a text document. The proliferation of the Web and the other Internet services in the past few years intensified the need for developing systems that can effectively recognize relevant information in the enormous amount of text that is available online. A variety of systems have been developed in the context of IE from online text (e.g. Freitag and Kushmerick, 1999; Sonderland, 1999; Freitag, 2000; Ciravegna, 2001; Califf and Mooney, 2003). The key idea behind combining a set of IE systems through stacking is to learn a common meta-level classifier, such as a decision tree or a naive-Bayes classifier, based on the output of the IE systems, towards higher extraction performance. On the other hand, a simpler approach is to vote on the predictions of different IE systems.

In order to apply voting and stacking to IE, the base-level classifiers should be normally replaced by systems that model IE as a classification task. The main problem, however, is that IE is not naturally a classification task (Thompson et al., 1999). A typical IE system is trained using a set of sample documents, paired with templates that are filled with relevant text fragments from the documents. IE could be mapped to a common classification problem by classifying almost every possible unbroken sequence of tokens (usually up to a predefined maximum length) that can be found within a document, as relevant or not (Freitag, 2000). This way of modelling the IE task, however, results in an enormous increase in the number of candidate text fragments, where only the small number of annotated fragments is considered as positive examples, while all the other fragments are considered as negative examples during training. Table 1 shows the examples that are constructed from a hypothetical text fragment within a page describing laptop products.

Text Fragment:	...processor 256 MB SDRAM...	
Positive Examples:	256 MB	
Negative Examples:	processor processor processor 256 MB ...	256 MB 256 MB SDRAM MB SDRAM ...

Table 1. An indicative example of recognizing an instance of the field *ram* (highlighted in bold) from a page that describes a laptop product and the set of examples it generates.

Although the size of the candidate text fragments can be somehow reduced by using various heuristics (Freitag, 2000), modelling IE in this manner, does not seem natural.

Alternative approaches of modelling the IE task exist in the literature. Systems like BWI (Freitag and Kushmerick, 1999), (LP)² (Ciravegna, 2001) and STALKER (Muslea et al., 2001), model IE as a boundary detection task. A boundary is the virtual space between two adjacent tokens. The task here is to recognize starting and ending token boundaries of relevant fragments within a document and then extract the enclosed content. In Table 1, the boundary between “” and “256” and the one between “MB” and “SDRAM” are the starting and ending index respectively, of the fragment “256 MB”. Some approaches (Freitag and McCallum, 1999, 2000; McCallum et al., 2000; Lafferty et al., 2001) model IE as the task of labelling the linear sequence of tokens that a text document is parsed into. Fragments consisting of (contiguous) tokens that

have been marked as relevant for a field (e.g. “256“, “MB”) are extracted. A variety of other approaches (e.g. Sonderland, 1999; Califf and Mooney, 2003) induce matching rules that extract whole fragments from a text document at runtime and fill the corresponding slots in the template.

This article initially introduces the idea of merging the templates filled by different IE systems into a single *merged* template, which facilitates the application of voting and stacking to IE. The merged template contains those text fragments that have been identified by at least one IE system, along with the individual predictions by the systems. Various voting schemes are then presented that rely either on the nominal or the probabilistic predictions of the IE systems that are available at the base-level.

A new stacking framework is then introduced that combines a wide range of base-level IE systems with a common classifier at the meta-level. Only the output of the IE systems is combined, i.e., the filled templates, which are *merged* into a single template, independently of how the instances that populate the templates were identified. In the new framework, only the meta-level data set consists of feature vectors that are constructed by the predictions of the IE systems, while the base-level data set consists of text documents, paired with filled templates. In contrast, both base-level and meta-level data sets in stacking for classification consist of feature vectors. An extension of the stacking framework for IE is also proposed that is based on using probabilistic estimates of correctness in the predictions of the IE systems.

Extensive experiments were conducted for comparing voting against stacking. Particular emphasis was given to analyzing the results obtained by voting and stacking with respect to how the base-level IE systems correlate in their output. Three well known IE systems were employed at the base-level, each drawn from a different learning paradigm: (LP)², a sequential covering rule-induction algorithm, Hidden Markov Models (HMMs), a finite-state approach to IE, and Boosted Wrapper Induction (BWI) that introduces the application of boosting to IE. A diverse set of classifiers were comparatively evaluated at the meta-level. Experiments were conducted on five collections of pages from five different domains.

The remainder of this article is structured as follows: Section 2 presents some background in the areas of voting, stacking and IE. Section 3 introduces the concept of the *merged* template and describes various voting schemes for IE. Section 4 describes the new stacking framework for IE. Section 5 describes the experimental design. Section 6 presents the results obtained by voting and stacking, and compares all IE systems at both base-level and meta-level. Section 7 explains the results obtained at the meta-level, with respect to the varying degree of correlation in the output of the base-level systems. Section 8 presents our conclusions, discussing potential extensions.

2 Background

Sections 2.1 to 2.3 provide background in the areas of voting, stacking and information extraction respectively.

2.1 Voting

The simplest way to combine the output of multiple classifiers is within a voting framework. Let $C^1 \dots C^N$ be the set of classifiers that are induced by training N different learning algorithms $L^1 \dots L^N$ on a data set D consisting of feature vectors. To classify a new instance at runtime, the classifiers $C^1 \dots C^N$ are queried for a class value and the class with the highest count is finally selected. This scheme is known as majority (or plurality) voting. Variations include weighted majority voting and voting using class probability distributions (Dietterich, 1997). In the former approach, each classifier’s vote is weighted by its accuracy, as measured by either using a holdout data set or the entire training data set by cross-validation. In the probabilistic approach, each classifier outputs a probability distribution vector over all relevant classes. For each class, the individual probability values are averaged (or summed) by all classifiers, and the class with the maximum value is finally selected.

Note that methods like *boosting* (Freund and Schapire, 1996) and *bagging* (Breiman, 1996) vote on a set $C^1 \dots C^N$ of classifiers that are generated by applying a single learning algorithm to N different versions of a given data set, rather than training N different algorithms.

2.2 Stacking

Section 2.2.1 presents stacking, while Section 2.2.2 describes some related work.

2.2.1 Definition

Wolpert (1992) introduced a novel approach for combining multiple classifiers, known as stacked generalization or stacking. The key idea is to learn a meta-level (or level-1) classifier based on the output of base-level (or level-0) classifiers, estimated via cross-validation as follows:

Define D a data set consisting of feature vectors, also referred to as level-0 data, and $L^1 \dots L^N$ a set of N different learning algorithms. During a J -fold cross-validation process, D is randomly split into J disjoint parts $D^1 \dots D^J$ of almost equal size. At each j th fold, $j=1 \dots J$, the $L^1 \dots L^N$ learning algorithms are applied to the training part $D \setminus D^j$ and the induced classifiers $C^1(j) \dots C^N(j)$ are applied to the test part D^j . The concatenated predictions of the induced classifiers on each feature vector x_i in D^j , together with the original class value $y_i(x_i)$, form a new set MD^j of meta-level vectors.

At the end of the entire cross-validation process, the union $MD = \cup MD^j$, $j=1 \dots J$ constitutes the full meta-level data set, also referred to as level-1 data, which is used for applying a learning algorithm L^M and inducing the meta-level classifier C^M . The learning algorithm L^M that is employed at meta-level could be one of the $L^1 \dots L^N$ or a different one. Finally, the $L^1 \dots L^N$ learning algorithms are applied to the entire data set D inducing the final base-level classifiers $C^1 \dots C^N$ to be used at runtime. In order to classify a new instance, the concatenated predictions of all base-level classifiers $C^1 \dots C^N$ form a meta-level vector that is assigned a class value by the meta-level classifier C^M . Figure 1(a) illustrates the cross-validation methodology, while Figure 1(b) illustrates the stacking framework at runtime.

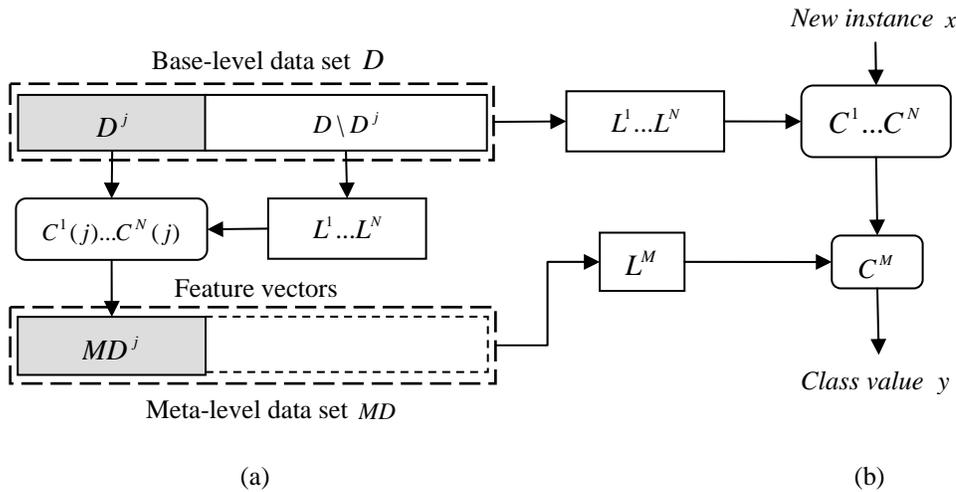


Figure 1. (a) Illustration of the J -fold cross-validation process for creating the meta-level data set. (b) The stacking framework at runtime.

2.2.2 Related Work

Research on stacking concerns two major issues, initially described as *black art* by Wolpert (1992). The first is the choice of classifiers at both base-level and meta-level that will lead to the best empirical results. The second issue, which has generally received more attention in the literature, concerns the combination of the predictions of the base-level classifiers and their mapping to attributes for the features vectors at the meta-level. Typical attributes that are used at the meta-level are the class predictions of the N base-level classifiers.

Chan (1996) experimented with various representations including the *class-attribute-combiner* scheme, where the class predictions of the base-level classifiers are appended with the attributes of the base-level vectors, together with the correct class for each vector. Chan (1996)

also experimented with an *arbiter* scheme, where a meta-classifier is only trained on a subset of the *base-level* vectors, in which the base-level classifiers disagree in their predictions. A *hybrid* scheme was also evaluated, in which a meta-classifier is only trained on a subset of the meta-level data set that follows the *class-attribute-combiner* scheme, where the base-level classifiers disagree in their predictions. Experimental results showed that the *class-attribute-combiner* is the best scheme. A slight improvement in the accuracy was obtained at meta-level over the best base-level results, but the differences were not measured as statistically significant.

Ting and Witten (1999) introduced a variant of stacking where each base-level classifier predicts a probability distribution vector over all classes, instead of predicting a single nominal value. The individual vectors by the N classifiers are concatenated, thus resulting in $N * Q$ attributes at meta-level, where Q the number of relevant classes. Ting and Witten (1999), suggested also the use of *multi-response linear regression* (MLR) for meta-level learning that proved to be highly effective. MLR is an adaptation of linear regression (Breiman, 1996a) which transforms the classification problem into Q different binary prediction problems: for each relevant class, a linear equation is constructed to predict one if the class value equals the class under consideration or zero otherwise.

Seewald (2003) suggested a modification of the approach described by Ting and Witten (1999), where different sets of meta-level features should be used for each of the Q binary prediction problems. In particular, only the probability values for the class under consideration should be used at meta-level, instead of concatenating the probability distributions of all classifiers, and thus reducing the number of meta-level attributes to N . Experimental results showed an improvement over stacking with probability distributions.

Džeroski and Ženko (2004) investigated the use of MLR in conjunction with class probability distributions augmented with an additional set of attributes that is based on the entropies of the class probability distributions and the maximum probability returned by each classifier. This scheme was found to perform better than using only probability distributions.

Stacking typically outperforms voting. However, voting does not involve cross-validation and the training of a meta-level classifier, and thus it is computationally cheaper than stacking.

2.3 Information Extraction

Sections 2.3.1 and 2.3.2 provide background on the task of Information Extraction (IE), while Section 2.3.3 describes an existing framework for combining multiple IE systems.

2.3.1 Definition

Let $\{f^1 \dots f^Q\}$ be a set of Q extraction *fields* for a particular domain of interest, and d a text document annotated by the domain experts with *instances* of those fields. A field *instance* is a pair $\langle t(s, e), f \rangle$, where $t(s, e)$ is a text fragment, with s and e be the boundaries of the fragment in a document's token table and $f \in \{f^1 \dots f^Q\}$ the associated field. A boundary has been defined above as the virtual space between two adjacent tokens. Define T a template that is filled with pairs $\langle t(s, e), f \rangle$. A field is typically a *target-slot* in template T , while $t(s, e)$ is a *slot-filler*. A field may also have multiple or no instantiations within a document. Table 2(a) shows a part of a Web page describing laptop products where the relevant text is highlighted in bold. Table 2(b) shows the hand-filled template for this page.

The Information Extraction (IE) task can be defined as follows: *given a new document d , find all possible instances for each relevant field within d and populate a template T* . This definition states that each field learning problem is considered in isolation, and thus modelled as a binary learning task: given a learning algorithm designed for IE, then for each relevant field $f \in \{f^1 \dots f^Q\}$, a *target concept* is learned that identifies relevant instances $\langle t(s, e), f \rangle$ within text. At runtime, all target concepts are applied separately to d and used to populate T .

An extended approach to IE is to study interactions among relevant fields, and thus grouping field instances into higher-level concepts, also referred to as *multi-slot* extraction (Sonderland, 1999). In this article we handle the simpler single-slot approach, which covers a wide range of IE tasks and motivated the development of a variety of learning algorithms (e.g. Freitag and Kushmerick, 1999; Freitag, 2000; Ciravegna, 2001; Califf and Mooney, 2003).

...**TransPort ZX**
 15" XGA TFT Display
 Intel Pentium III 600 MHZ 256k Mobile processor
 256 MB SDRAM up to 1GB
 40 GB hard drive (removable)
 ...

(a)

T			Short description for field f
$t(s,e)$	s,e	Field f	
TransPort ZX	47, 49	model	Name of the laptop's model
15"	56, 58	screenSize	Size of the laptop's screen
TFT	59, 60	screenType	Type of laptop's screen
IntelPentium III	63, 67	procName	Name of the laptop's processor
600 MHZ	67, 69	procSpeed	Speed of the laptop's processor
256 MB	76, 78	ram	The RAM capacity of the laptop
40 GB	86, 88	HDcapacity	The hard disk capacity of the laptop

(b)

Table 2. (a) Part of a Web page describing laptop products (b) The hand-filled template for this page.

2.3.2 Related Work

The IE task from free text has been the focus of the Message Understanding Conferences (e.g. DARPA 1995, 1996). On the other hand, the advent of the Web intensified the need for developing systems that help people to cope with the large amount of text that is available online. Systems that perform IE from online text, should generally meet the requirements of low cost and high flexibility in development, and adaptation to new domains. MUC-level systems fail to meet those criteria, in addition to the fact that the linguistic analysis performed for free text does not exploit the extra-linguistic information (e.g. HTML/XML tags, layout format) that is available in online text. Therefore, this type of system has not found wide applicability in the context of online sources.

As a result, less linguistically intensive approaches have been developed for IE on the Web using *wrappers*, which are sets of highly accurate rules that extract a particular resource's content. The manual development of wrappers (Chawathe et al., 1994) has proved to be a time-consuming task, requiring a high-level of expertise. Machine-learning techniques that learn wrappers for IE, either using supervised learning (e.g. Kushmerick, 1997; Muslea et al., 2001; Cohen et al., 2002) or unsupervised learning (e.g. Crescenzi et al., 2001; Chang and Lui, 2001), have been designed to handle highly structured collections of Web pages, such as telephone directories and product catalogues. Those approaches, however, fail when the text type is less-structured, which is also common on the Web.

Recent effort on *adaptive* IE (Ciravegna, 2001; Ciravegna and Lavelli, 2003), motivates the development of IE systems that can handle different text types, from rigidly structured to almost free text -where common wrappers fail- including mixed types. For example, the algorithms presented in (Sonderland, 1999; Ciravegna, 2001; Califf and Mooney, 2003) learn IE rules that exploit shallow natural language knowledge and thus can be applied to less structured text. The BWI algorithm (Freitag and Kushmerick, 1999) relies on a method called *boosting* (Freund and Schapire, 1996) for improving the extraction performance of the learned IE rules, which allows the applicability of the algorithm to less structured text. Hidden Markov modelling (Rabiner, 1989) is a powerful statistical learning technique that has found wide applicability in IE from both structured and unstructured text (Seymore et al., 1999; Freitag and McCallum, 1999, 2000).

In this article we focus on adaptive IE systems and investigate how their performance can be further improved, by combining their output at meta-level. The presence of the token boundaries s and e is essential, as we will show, for combining different IE systems. In some cases (e.g.

Ciravegna, 2001), information about token boundaries is implicitly represented by inserting appropriate start and end XML tags within documents, i.e. $\langle f \rangle$ and $\langle /f \rangle$ tags for each relevant field $f \in \{f^1 \dots f^{\ell}\}$. A template such as the one in Table 2(b) can then be easily constructed. Thus, we assume in the remaining of this article that we deal with templates that include information about token boundaries, as in Table 2(b), which is a realistic assumption, since such information is typically available.

2.3.3 Multistrategy Learning

Despite the growing interest in combining machine learning algorithms and the application to some natural language parsing tasks such as part-of-speech tagging (Halteren et al., 2001) and word-sense disambiguation (Florian et al., 2002), this topic has received little attention by the IE community. The only relevant work is described in (Freitag, 2000), where the IE task was modelled as a classification one, using a set of four base-level systems, which were then combined by multistrategy learning.

The term *multistrategy learning* generally refers to the combination of multiple learning approaches under a single algorithm and was mainly used for combining inductive with analytical learning (Michalski and Tecuci, 1994). Domingos (1996) used the term *empirical multistrategy learning* for distinguishing the case where all learning components are inductive. The basic notion behind empirical multistrategy learning was to design a new complex algorithm that heuristically combines a set of learning components by requiring, however, implementation details of each individual component. For example, the RISE algorithm in (Domingos, 1996) unifies an instance-based learner with a rule-based learner under a new implementation, aiming to overcome the limitations of both approaches.

Stacking combines a set of multiple learning components in a more loose fashion, by only learning to integrate their output at meta-level, and thus treating them as *black boxes*. Despite its simplicity, stacking offers the advantage of being highly extensible to more algorithms at both base-level and meta-level, regardless of their internal structure. On the other hand, voting is the simplest form of loosely integrating the output of multiple components.

Freitag (2000) followed the voting paradigm in the context of IE, which he called “multistrategy learning for IE” and it is based on using probabilistic estimates produced by the IE systems. Specifically, for each relevant field $f \in \{f^1 \dots f^{\ell}\}$, the confidence scores that are produced by the base-level systems are mapped into probabilistic estimates of correctness in the predictions of the systems. In case of more than one predictions of the field f for a fragment $t(s, e)$, a combined probability is estimated for $\langle t(s, e), f \rangle$ using (1).

$$P^c = 1 - \prod_j (1 - p^j), \quad (1)$$

where P^c is the combined probabilistic estimate that the text fragment $t(s, e)$ belongs to the field f , and p^j the probabilistic estimate that some IE system E^j has predicted the field f for $t(s, e)$. Actually, P^c measures the probability that at least one of those IE systems that have predicted the field f for $t(s, e)$, has predicted correctly, which equals the probability that not all predictions for f are wrong.

Finally, the constraint of “one per document” (OPD) is imposed on some fields. This means that only one instance of OPD fields is allowed for each page. For example, a page in the domain of computer science (CS) courses should describe only one course, and thus should contain only one instance of the field *course title*. Therefore, if more are identified, then the one with the highest (combined) probability is selected, while all others are rejected. The example in Table 3 illustrates the usefulness of the OPD constraint.

Assuming in Table 3 that one hypothetical IE system predicts “256 MB” and “1 GB” as ram instances, while another system predicts only “256 MB” as *ram*. The combination of the two systems, under the OPD constraint, produces a single instance $\langle \text{“256 MB”}, \text{ram} \rangle$, the one that has the highest combined probability.

$t(s,e)$	Probability by the first system	Probability by the second system	Combined probability
256 MB	0.4	0.5	0.7
1GB	0.6	-	0.6

Table 3. Combining the predictions of two hypothetical IE systems for a “ram” instance. Supposing that the correct instance is <“256 MB”, ram >

The OPD field constraint, though useful in certain cases, is restrictive for IE in general and does not hold for all relevant fields. For example, a Web page may describe more than one laptop products, and thus more than one *ram* instances may exist. The OPD constraint was also applied for fields that allow many instances per page, without significant loss of performance¹. For example, a page may rarely describe more than one CS courses. This approach is still restrictive for IE, since a Web page very often describes more than one laptop products.

Finally, converting confidence scores to probabilistic estimates takes place by validating the performance of each IE system on a hold-out set, or by cross-validation in the entire training data set and using a form of regression modelling which is described in more detail by Freitag (2000). The motivation behind mapping confidence scores to probabilistic estimates is that confidence scores are not always reliable, since incorrect matches may be assigned high scores. Thus voting on different IE systems using confidence scores may not always be reliable. The correlation among confidences and probabilities has been also investigated by Kauchak et al. (2004) in the context of the BWI algorithm for IE, and found to be weaker for more difficult IE tasks, e.g. from free-text domains.

In the remainder of this article, the term “multistrategy learning” will be used to refer to the work in (Freitag, 2000).

3 Voting for Information Extraction

Section 3.1 presents an example of combining IE systems. The concept of the *merged* template is introduced, which is important for combining different IE systems either through voting or stacking. Various voting schemes for IE are then presented in Sections 3.2 and 3.3, against which the performance of stacking for IE will be compared.

3.1 Example of Combining Different Systems – The Merged Template

Let $L^1 \dots L^N$ be a set of N learning algorithms, designed for IE, which are given a corpus D of training documents, annotated with relevant field instances. The algorithms $L^1 \dots L^N$ typically generalize from the training corpus, towards a set of pattern-matching extraction rules. Define $E^1 \dots E^N$ the corresponding set of IE systems that exploit the acquired knowledge, to identify relevant instances in new documents. Each trained IE system consists of a set of target concepts that have been learned for the relevant fields. Finally, define $T^1 \dots T^N$ a set of templates for a document d , populated by $E^1 \dots E^N$ respectively with relevant field instances.

We suggest in this article that a *merged* template can be constructed from $T^1 \dots T^N$ as follows: all text fragments $t(s,e)$ identified by $E^1 \dots E^N$ in $T^1 \dots T^N$ are inserted to an initial pool. Duplicate fragments are removed: two fragments differ if either their start or end boundary differs. For the remaining *distinct* fragments, the fields predicted by $E^1 \dots E^N$ are collected and inserted together with the correct field in the template. If some IE system does not predict a field for a text fragment, then the corresponding cell in the merged template is empty. If a text fragment does not exist in the hand-filled template, then the corresponding cell in the last column is also empty. Table 4 shows an illustrative example of a merged template that has been constructed by the output T^1, T^2 of two IE systems E^1, E^2 , for the page of Table 2(a).

¹ This is a remark by Dayne Freitag, based on personal contact.

s, e	$t(s, e)$	Output by E^1	Output by E^2	Correct field
47, 49	TransPort ZX	model	manuf	model
56, 58	15"	screenSize	-	screenSize
59, 60	TFT	screenType	screenType	screenType
63, 66	IntelPentium	-	procName	-
63, 67	IntelPentium III	procName	-	procName
67, 69	600 MHz	procSpeed	procSpeed	procSpeed
76, 78	256 MB	ram	ram	ram
81, 83	1 GB	ram	HDcapacity	-
86, 88	40 GB	-	HDcapacity	HDcapacity

Table 4. Merged template, based on the output of two IE systems. Each entry corresponds to a text fragment that has been identified by at least one system.

Examining Table 4, we note some disagreement in the predictions of the two systems. For two text fragments (“TransPort ZX”, “1GB”) the predicted fields by E^1 and E^2 differ. Comparing to the hand-filled template of Table 2(b), we conclude that “TransPort ZX” has been correctly identified as *model* only by E^1 , while E^2 identified the same fragment as *manuf* (the manufacturer of the laptop). On the other hand, the fragment “1GB” does not exist in the hand-filled template. Therefore, the fields predicted by the two systems for this fragment are false. Furthermore, some text fragments have been identified by only one of the two IE systems. The fragment “15” has been identified only by E^1 , while the fragment “40 GB” has been identified only by E^2 . The fields predicted for both fragments are correct.

Examining again Table 4, we wonder whether we can exploit, at some higher level, the disagreement in the predictions of the different IE systems, aiming to achieve superior extraction performance. The desirable result is to automatically fill the last column in the merged template of Table 4 with the correct fields. In other words, we would like to assign the correct field to each text fragment that has been identified by at least one base-level system.

3.2 Majority Voting

A simple idea for combining the predictions of different IE systems is to use majority voting: for each entry in the merged template, we count the predicted fields by the available systems and select the field with the highest count. In the case of a tie, a random selection is typically performed among even fields.

Note that Table 4 contains missing values, reflecting the natural fact that some system may not have predicted a field for a text fragment that has been identified by another system. The significance of missing values has to be carefully considered. For example, if some system predicts an incorrect field f for a text fragment $t(s, e)$, while the remaining systems do not predict any field at all, then ignoring missing values during voting harms precision, since the incorrect field is returned. An alternative is to record a missing value as “false”, providing evidence that no field should be predicted for $t(s, e)$. If the value with the highest count is “false” then no field is assigned to $t(s, e)$. If, however, f is the correct field for $t(s, e)$, interpreting the missing predictions by the remaining systems as “false” values harms overall extraction performance, since the correct field is rejected.

Therefore, two different settings of majority voting are defined, depending on whether missing values are ignored or encoded as “false” values that indicate rejection of prediction.

3.3 Voting Using Probabilities

The voting with probabilities scheme that is presented in this section shares many features with multistrategy learning, as described in (Freitag, 2000) and was briefly outlined in Section 2.3.3. Both schemes share the same method for mapping confidence scores to probabilistic estimates and the same Equation (1) that estimates the combined probability of correctness for an instance $\langle t(s, e), f \rangle$. However, the two schemes differ in how they model the IE task.

Multistrategy learning considers each field in isolation during combination and relies on the OPD constraint for improving the extraction accuracy, as demonstrated by the example of Table

3. On the other hand, voting using probabilities takes place on a merged template, like the one in Table 4, while no OPD assumption is required for any relevant field. This allows the case of contradictory field predictions among different systems during combination, as demonstrated in Table 4, where the fragment “1GB” has been identified as *ram* by the first system and as *HDcapacity* by the second one. The field with the highest probability should be selected. Multistrategy learning for IE ignores contradictory field predictions.

In Section 3.2 two different settings of majority voting were defined, depending on whether absence of prediction by some system for a text fragment, i.e. a missing value in the merged template, is ignored or encoded as “false” that indicates rejection of prediction. The problem here is that there is no probability for “false”. If we assume that “false” corresponds to probability 1, then voting will lead to spurious results. Thus, two different settings for voting using probabilities are defined as follows:

In the first setting, missing values are ignored, similar to the first setting of majority voting. Given a fragment $t(s,e)$, the field f with the highest probabilistic estimate by those systems that have predicted a field for $t(s,e)$ is returned. In the second setting, however, a constraint is imposed on whether f should be accepted or not. If the probability that is attached to f is less than 0.5, then f is rejected. Otherwise, f is returned, as in the first setting. The motivation behind using this constraint, is that if f has been predicted with low degree of confidence by the base-level systems, then it should not be accepted. The value of 0.5 is the natural choice of a threshold for deciding whether f should be accepted or not.

4 Stacked Generalization for Information Extraction

This section starts with the motivation for performing learning, rather than simply voting. Then a new stacking framework for IE is presented, along with an extension that relies on using probabilistic estimates on the output of the base-level systems.

4.1 Motivation for Performing Learning

Examining the merged template of Table 4, we wonder whether we can *learn* to predict the correct field, based on the fields predicted by the available systems, rather than simply voting. A simple motivation for preferring learning, rather than voting, is that the latter cannot handle situations where most of the systems make an error. For example, if a system correctly predicts *ram* for the hypothetical fragment “1,5 GB”, while the other systems erroneously predict *HDcapacity*, then voting chooses the latter value. Therefore, it would be desirable to perform learning in order to induce a rule of the form: *if the first IE system predicts “ram” and the other systems predict “HDcapacity”, then the correct field is “ram”*.

In order to train a common classifier, a set of feature vectors should be provided as training data. The idea suggested in this article is to create a feature vector for every row entry of the merged template, i.e. for each text fragment that has been identified by at least one base-level system. Table 5 shows the new feature vectors created by the merged template of Table 4.

s,e	$t(s,e)$	Feature vectors		
		Output by E^1	Output by E^2	Class
47, 49	TransPort ZX	model,	manuf,	model
56, 58	15"	screenSize,	?,	screenSize
59, 60	TFT	screenType,	screenType,	screenType
63, 66	IntelPentium	?,	procName,	false
63, 67	IntelPentium III	procName,	?,	procName
67, 69	600 MHz	procSpeed,	procSpeed,	procSpeed
76, 78	256 MB	ram,	ram,	ram
81, 83	1 GB	ram,	HDcapacity,	false
86, 88	40 GB	?,	HDcapacity,	HDcapacity

Table 5. Feature vectors created by the merged template of Table 4.

Absence of prediction by an IE system is indicated by “?”. If a text fragment does not exist in the hand-filled template, the class attribute of the corresponding vector takes the value “false” that indicates rejection of prediction. At runtime, the class value is to be assigned by the classifier.

Having specified the format of the feature vector, the remaining issue is to construct the full set of vectors for training the meta-level classifier using cross-validation, as described for the stacking framework in Section 2.2. However, in IE we deal with collections of text documents, annotated with relevant instances, rather than feature vectors. This disparity between base-level and meta-level data sets is handled by sampling from documents during cross-validation, rather than from feature vectors as in stacking for classification.

4.2 Stacking Using Nominal Values

The key idea behind stacking for IE, is to learn a meta-level classifier based on the output of base-level systems via cross-validation as follows:

At the j th fold, $j = 1..J$, of cross-validation, the N learning algorithms $L^1..L^N$ are trained on the document set $D \setminus D^j$ and the induced IE systems $E^1(j)..E^N(j)$ are applied to the test set D^j . For each document d in D^j , let $T^1..T^N$ be the populated templates by $E^1(j)..E^N(j)$ respectively. A merged template MT is assembled from $T^1..T^N$, as shown in Section 3.1. A new feature vector is produced for each entry in the merged template, which is added to the meta-level data set MD^j . At the end of the cross-validation process, the union $MD = \cup MD^j$ constitutes the full meta-level data set, which is used by a learning algorithm L^M to train the meta-level classifier C^M . Finally, the N learning algorithms are applied to the entire data set D , inducing the base-level systems $E^1..E^N$ to be used at runtime.

Figure 2 presents an algorithmic description of the new stacking framework for IE. The vectors in the new meta-level data set MD belong to $Q+1$ classes, where Q is the number of relevant fields in the domain of interest plus the value “false”. A vector classified as “false” indicates that the corresponding fragment $t(s,e)$ does not exist in the hand-filled template, and thus the available base-level systems should not have predicted a field for it (for example the fragment “1 GB” in Table 5).

```

procedure stacking_for_IE ( $D, J, L^1..L^N, L^M$ ) begin
     $D^1..D^J$  = partition of  $D$  into  $J$  document collections of almost equal size
    for  $j = 1$  to  $J$  do begin
         $MD^j = \{\}$ 
        for  $i = 1$  to  $N$  do  $E^i(j)$  = the system obtained by training  $L^i$  on  $D \setminus D^j$ 
        foreach document  $d$  in  $D^j$  do begin
            for  $i = 1$  to  $N$  do  $T^i$  = the template, populated by applying  $E^i(j)$  to  $d$ 
             $MT = \text{create\_merged\_template}(d, T^1..T^N)$ 
            foreach entry, i.e., for each distinct  $t(s,e)$ , in  $MT$  do begin
                for  $i = 1$  to  $N$  do  $f^i \in \{f^1, \dots, f^Q, "?\}$  = the field by  $E^i(j)$  for  $t(s,e)$ 
                 $f \in \{f^1, \dots, f^Q, \text{false}\}$  = the correct field for  $t(s,e)$ 
                 $MD^j = MD^j \cup \text{vector} \langle f^1, \dots, f^N, f \rangle$ 
            end
        end
    end // end of cross validation
     $MD = \cup MD^j, j=1..J$ 
     $C^M$  = meta-level classifier obtained by applying  $L^M$  on  $MD$ 
    // Train the base-level IE systems
    for  $i = 1$  to  $N$  do  $E^i$  = the base-level system obtained by training  $L^i$  on  $D$ 
end
    
```

Figure 2. The new stacking framework for information extraction.

The key difference among stacking for IE and common stacking is that cross-validation operates on text documents paired with hand-filled templates, instead of feature vectors labelled with class values. This removes the constraint of performing common classification at base-level, thus allowing the application of stacking to IE. The base-level algorithms $L^1 \dots L^N$ are designed for IE, while the learning algorithm L^M that is applied at meta-level is designed for classification, and thus cannot be one of $L^1 \dots L^N$ as in stacking for classification.

The size of the meta-level data set is not a-priori known in stacking for IE, unlike common stacking where there is a one-to-one correspondence between base-level and meta-level vectors. Here, the size of the meta-level data set is determined by the output of the base-level systems, i.e. by the individual templates $T^1 \dots T^N$ that assemble a merged template, as shown in Figure 2.

The one-to-one correspondence between base-level and meta-level features vectors in common stacking, results in identical class values at both base-level and meta-level. In IE, however, a text fragment that is relevant to our task may not have been identified by any of the available base-level systems. In that case, there is no possibility of identifying that fragment at meta-level, since no feature vector is created and thus resulting in loss of information. This observation suggests that IE systems biased towards *recall* (percentage of the annotated field instances that were identified), should be generally preferred for combination, expecting to reduce the loss of information at meta-level.

Moreover, a meta-level vector in common stacking for classification does not contain missing values, since each base-level classifier predicts a nominal class value or a probability distribution over the relevant classes. On the other hand, a meta-level vector in stacking for IE may contain missing values, as shown in Table 5, since some system may not have predicted a field for a fragment that has been identified by another system. In correspondence to majority voting, missing values in Table 5 can be handled by recording them as “false”, which indicates rejection of prediction. In that case, all attribute values, should share the same set of values $\{f^1 \dots f^Q, false\}$, thus replacing the set $\{f^1 \dots f^Q, "?"\}$ in Figure 2.

Another issue in stacking for IE is the choice of J in the J -fold cross-validation process depicted in Figure 2. The choice of J usually depends on the size of the training data and the computational cost of training. The choice of J also affects the difference in the number of documents for training the base-level systems and the meta-level classifiers. According to Figure 2, the base-level systems $E^1 \dots E^N$ that will be used at runtime are retrained on the entire collection D of training documents. On the other hand, the meta-level vectors on each j th fold are created by the predictions of the systems $E^1(j) \dots E^N(j)$, as trained on a lower proportion of the training documents $D \setminus D^j$. The larger the value of J , the smaller the size of D^j and therefore the smaller the difference between the size of the training data set $D \setminus D^j$ for $E^1(j) \dots E^N(j)$ and the complete data set D .

To illustrate this point, for a collection of 40 training documents, the final base-level systems $E^1 \dots E^N$ are trained on the full data set. If we assume a five-fold cross validation process, then on each fold j , the $E^1(j) \dots E^N(j)$ are trained on 32 documents. An alternative is to use a higher value for J , suffering however a higher computational cost.

4.3 Stacking at Runtime

Given a new document d , the systems $E^1 \dots E^N$ are used to identify relevant field instances and fill the corresponding templates $T^1 \dots T^N$. A merged template is then created from $T^1 \dots T^N$. For each row entry in the merged template, i.e., for each distinct $t(s, e)$, a feature vector is created by the predicted fields of $E^1 \dots E^N$ for $t(s, e)$ (absence of prediction by an IE system is indicated by “?” or “false”). The new vectors are finally classified by the meta-level classifier C^M into $Q + 1$ predefined categories $\{f^1 \dots f^Q, false\}$. If a vector is classified into one of the relevant fields $f^1 \dots f^Q$ then the corresponding instance $\langle t(s, e), f \rangle$ is inserted in the final template for d . Otherwise (“false” prediction) the entry is excluded from the final template. For example, if we assume that the class column in Table 5 has been filled by C^M , then for the two vectors that have been classified as “false”, the corresponding entries will be excluded from the final template. At runtime, the stacking framework for IE is graphically depicted in Figure 3.

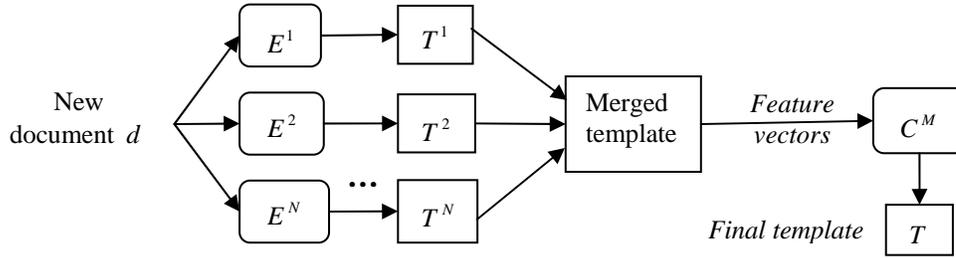


Figure 3. The stacking framework for information extraction at runtime.

According to Figure 3, the input to the systems is a new text document while the output is the corresponding filled template. In contrast, both input and output in the runtime stacking for classification architecture of Figure 1(b), consist of a single feature vector.

4.4 Stacking Using Probabilities

A straightforward extension of stacking with nominal values is to rely on the confidence scores by the base-level systems that have been converted into probabilistic estimates of correctness. The new framework is described as follows:

- Instead of predicting one of the Q relevant fields for each fragment $t(s, e)$, each system generates a confidence score c^k for the predicted field f^k . This is modelled by a Q -element vector that contains zero values, except for the k th position where c^k appears, i.e., $\langle 0, \dots, c^k, \dots, 0 \rangle$. If a system does not predict a field, all elements are zero.
- Each vector is converted to a new one $\langle 0, \dots, p^k, \dots, 0 \rangle$, where p^k is a probabilistic estimate that corresponds to c^k and reflects the probability of correctness of the prediction. The conversion process is described in more detail in (Freitag, 2000).
- Finally, the output vectors by $E^1 \dots E^N$ for $t(s, e)$ form a single vector of $N * Q$ elements, appended by the correct field, according to the hand-filled template.

Table 6 shows an illustrative example of the new feature vectors at the meta-level, using probabilistic estimates of correctness.

s, e	$t(s, e)$	<i>Feature vectors using probabilistic estimates</i>		
		Output by E^1	Output by E^2	Class
47, 49	TransPort ZX	0, 0, 0.92, 0, 0, 0, 0, 0, 0,	0, 0.34, 0, 0, 0, 0, 0, 0, 0,	model
56, 58	15"	0, 0, 0, 0, 0, 0, 0.83, 0,	0, 0, 0, 0, 0, 0, 0, 0, 0,	screenSize
59, 60	TFT	0, 0, 0, 0, 0, 0, 0, 0.85,	0, 0, 0, 0, 0, 0, 0, 0.91,	screenType
63, 66	IntelPentium	0, 0, 0, 0, 0, 0, 0, 0,	0, 0, 0, 0.61, 0, 0, 0, 0,	false
63, 67	IntelPentium III	0, 0, 0, 0.67, 0, 0, 0, 0,	0, 0, 0, 0, 0, 0, 0, 0,	procName
67, 69	600 MHz	0, 0, 0, 0, 0.82, 0, 0, 0,	0, 0, 0, 0, 0.79, 0, 0, 0,	procSpeed
76, 78	256 MB	0, 0, 0, 0, 0, 0.91, 0, 0,	0, 0, 0, 0, 0, 0.77, 0, 0,	ram
81, 83	1 GB	0, 0, 0, 0, 0, 0.55, 0, 0,	0.89, 0, 0, 0, 0, 0, 0, 0,	false
86, 88	40 GB	0, 0, 0, 0, 0, 0, 0, 0,	0.65, 0, 0, 0, 0, 0, 0, 0,	HDcapacity

Table 6. The new meta-level vectors using probabilistic estimates of correctness.

The same vector representation used in Table 6 was also used in the extension of stacking for classification proposed in (Ting and Witten, 1999). The only difference is that class (or field) probability distributions, as they suggest, are not typically produced by IE systems. Given a text fragment $t(s, e)$, either a field f is predicted for $t(s, e)$ or no field is predicted at all. Thus, except for the vector elements that correspond to the predicted fields, all other values are set to zero, as shown in Table 6.

In stacking with probabilities, a missing value is indicated by a vector of which all elements take zero values. Missing values can be handled by augmenting the output of each of the N base-level systems with an additional attribute, indicating the probability for “false”. The total

number of meta-level attributes will be then $N(Q+1)$. The probability value of the extra attribute will be complementary to the value of the non-zero element of the vector. For example, in the first row entry of Table 6, the value of the extra attribute for the system E^1 will be 0.08, while for each absent prediction the extra attribute takes the value “1”. The significance of handling the missing values in stacking is empirically evaluated by comparing the performance of the trained classifiers over the new vectors against the trained classifiers over the vectors with missing values.

5 Experimental Setup

We have performed extensive experiments in five real-world domains, using well-known algorithms at both base-level and meta-level. The primary target was to determine whether stacking provides added value over the base-level systems and voting in the examined domains. Therefore, we comparatively evaluated all combination methods (voting and stacking) for IE, as described in Sections 3 and 4, while also comparing against the best base-level system for each domain of interest. Since the success of stacking relies on the disagreement in the output of the base-level components, we were particularly interested in how stacking behaves with respect to the diversity in the output of the base-level systems, and how that compares to voting.

5.1 Domains

Experiments were conducted using five collections of text documents from five different domains. The first two collections consist of 101 Web pages describing computer science (CS) courses and 96 Web pages describing research projects respectively, and were constructed in the context of the WebKB project (Craven et al., 1998). Both collections were hand-filled for three and two extraction fields respectively: *crsNumber*, the official number of the course (for example, “CS 305”), *crsTitle*, the title of the course (for example, “Operating Systems”), *crsInst*, the name of the course instructor, *projTitle*, the title of the project (for example, “WebKB”) and *projMember*, the name of a project member.

The third collection consists of 50 Web pages, describing laptop products that were collected from various vendor sites. A total of 19 fields were hand-filled, including the manufacturer of the laptop, model name, processor name, speed, ram, hard disk capacity, etc. This collection was constructed in the context of building a shopping comparison agent² that visits various vendor sites, extracts laptop descriptions and presents the results to user.

The last two collections consist of 300 pages describing job announcements from the *austin.jobs* newsgroup at Austin and 485 pages describing seminar announcements from the Carnegie Mellon University, respectively. Both collections were obtained from RISE (1998) and have been widely used in information extraction research. A total of 17 fields were hand-filled for job announcements, including the title of the available position, the salary, the name of the company, the identifier code of the announcement, etc. Four fields were hand-filled for seminar announcements: *stime*, the starting time of the seminar, *etime*, the ending time of the seminar, *speaker*, the speaker’s name and *location*, the location of the seminar.

Note that the available hand-filled templates for job announcements (Califf and Mooney, 2003) do not contain information about the starting and ending token boundaries of the annotated instances, which is however essential for combining different IE systems. Therefore, the entire corpus was re-annotated, using the available hand-filled templates as a guide, so that the new templates include token boundary information, as the one in Table 2(b).

Finally, the HTML tags in the three Web domains were not omitted during the training of the base-level systems, but appropriately tokenized, including their attributes and values. For example, the stream `<td valign="top">` corresponds to the subsequence “*td_start_tag*”, “*attrib_valign*”, “*value_top*” in the token table of a Web page.

² CROSSMARC, R&D project, IST-2000-25366, <http://www.iit.demokritos.gr/skel/crossmarc>

5.2 Base-level Information Extraction Systems

At the base-level we employed three systems that are well known in the literature for information extraction: the (LP)² system, the BWI system and a HMM-based IE system.

The *Learning Pattern by Language Processing* or (LP)² system implements a sequential covering algorithm that learns symbolic pattern-matching rules for IE. For each interesting field, a set of start-rules and another one of end-rules are induced that identify the starting and ending boundaries respectively of the relevant instances. Shallow natural language knowledge is used during the induction process such as lexical information (e.g. *capitalized*, *numerical*), part-of-speech tagging (for example, *noun*) and stemming information. Additional *contextual* and *correction* rules are learned that improve the performance of the previously induced rules. Each instance $\langle t(s,e), f \rangle$ that is recognized at runtime is assigned a confidence score $LS = \text{wrong} / \text{matched}$, where *wrong* is the number of erroneous matches, as estimated during training, of the rules that matched $t(s,e)$, and *matched* is the total number of matches by the rules. The lower this score, the higher the confidence attached to the instance. A detailed description of (LP)² can be found in (Ciravegna and Lavelli, 2003).

The *Boosted Wrapper Induction* or BWI system learns also symbolic starting and ending pattern-matching rules for each relevant field. Each rule is assigned a confidence score according to a boosting methodology, which is described in more detail in (Freitag and Kushmerick, 1999). The induced rules exploit lexical information (e.g. *capitalized*, *numerical*). Each instance $\langle t(s,e), f \rangle$ recognized at runtime is assigned the product of confidences of the start and end rules that match $t(s,e)$. The more rules match $t(s,e)$, the higher is the value of the score that is assigned to the instance. A comprehensive analysis of the performance of BWI in a variety of IE tasks can be found in (Kauchak et al., 2004).

Finally, our HMM-based IE is inspired by work described in (Freitag and McCallum 1999; Seymore et al., 1999), whereby a separate HMM is trained for each relevant field. The entire training page is probabilistically modelled by the HMM, by assuming that the first token of the page is emitted by the initial state of the model, then transitioning to the next state that emits the second token, etc. until the ending token of the page is emitted. Special *prefix*, *suffix* and *target* states model the immediate prefix, suffix and the internal structure of the relevant instances respectively. Inducing a HMM for each field involves the calculation of the state-transition and token-emission probabilities over all training pages, based on simple ratios of counts. The *Viterbi* algorithm is used at runtime to identify relevant instances and assign to them a confidence score. More details on how HMMs assign confidence scores for IE can be found in (Sigletos, 2005). An excellent tutorial on HMMs can be found in (Rabiner, 1989).

5.3 Meta-level Algorithms for Classification

At meta-level, we employ the following algorithms, implemented in the WEKA data mining platform (Witten and Frank 2000):

- *J48*, the well known C4.5 (Quinlan, 1993) decision tree algorithm.
- *NaiveBayes*, the well known Naïve Bayes classifier (John and Langley, 1995).
- *IB1*, the 1-nearest-neighbor algorithm.
- *Multi-response linear regression (MLR)*, an adaptation of least-squares linear regression (Breiman, 1996a).
- *SMO*, a fast implementation of Support Vector Machines (Platt, 1999).
- *LogitBoost*, an implementation of the corresponding algorithm (Friedman et al., 1999), using decision stumps as weak-classifier.

Most of these algorithms have already been evaluated as meta-level classifiers in recent studies for stacking (Ting and Witten, 1999; Seewald, 2003; Džeroski and Ženko, 2004).

5.4 Evaluation Methodology and Metrics

For the evaluation, cross-validation was used to obtain an unbiased estimate of performance over unseen data. For the domains of laptop products and job announcements, the corpus was randomly split into 5 equally populated parts. At each fold, a different part of pages was kept for

evaluation, and the pages in the remaining four parts were used in order to induce the base-level systems and the meta-level classifiers. Results on the test parts were averaged over all 5 folds. Notice that within the training part at each of the 5 folds, a separate 5-fold cross-validation procedure was used in order to create the meta-level set of feature vectors and thus train the classifiers, as described in Section 4.2.

For the two WebKB domains and seminar announcements, a different evaluation methodology was followed that was also applied in (Freitag, 2000). Each corpus was randomly split into 2 parts of almost equal size. The first part was used to induce the base-level systems and the meta-level classifiers, while the second part was used for evaluation. An internal 3-fold cross-validation process was followed in the training part in order to collect the meta-level set of feature vectors and then train the classifiers. The whole process was repeated 5 times, averaging the results at the end. Moreover, the constraint of “one instance per document” (OPD) was applied for the fields *crsNumber* and *crsTitle* in CS courses, for the field *projTitle* in research projects, and for all four fields in seminar announcements, towards an objective comparison against multistrategy learning for information extraction and the results presented in (Freitag, 2000). Therefore, whenever two or more instances of an OPD field were present within a page, only the instance with the highest score was selected.

Three metrics were used for measuring the performance: *precision* (P), the percentage of the identified field instances that are correct, *recall* (R), the percentage of the annotated field instances (in the hand-filled templates) that were identified, and finally $F1$, the harmonic mean of recall and precision defined as $F1 = 2RP / (R + P)$. Our definition of recall and precision is equivalent to *micro-average recall* and *micro-average precision* (Sebastiani, 2002), formally defined as:

$$P = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^Q TP^i}{\sum_{i=1}^Q (TP^i + FP^i)}, \quad R = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^Q TP^i}{\sum_{i=1}^Q (TP^i + FN^i)},$$

where TP^i is the number of instances of the field $f^i \in \{f^1, \dots, f^Q\}$ that have been correctly identified (*true positive*), FP^i is the number of f^i instances that have been incorrectly identified (*false positive*), and finally FN^i is the number of f^i instances that were not identified (*false negative*). Choosing micro-average metrics allows for an objective overall comparison among different systems, by considering all target instances by all relevant fields. Statistical significance in the conducted comparisons was evaluated using the well-known paired t -test (Dietterich, 1998) with a significance level of 95%.

6 Results and Comparisons

The results obtained by all base-level systems in the domains of interest are initially presented in this section, while also investigating whether any improvement in the best results for each domain is possible at meta-level. Then, the meta-level data is analyzed, in order to determine whether and how the predictions of the base-level systems are correlated. This study is intended to serve as a basis for a comparative evaluation of voting against stacking. Then all combination methods are comparatively evaluated, while also comparing against the best base-level results. More detailed analysis of the experimental results is provided in Section 7.

6.1 Results of Base-level

Table 7 shows the $F1$ scores (%) obtained by the base-level systems in the domains of interest. Only the highest $F1$ score for research projects was measured as statistically insignificant. Appendix B.1 shows the scores obtained in all three measures of performance.

	CS courses	Projects	Laptops	Jobs	Seminars
BWI	51.30	60.75	62.26	80.01	83.09
HMM	59.39	61.64	63.81	75.71	79.20
(LP) ²	65.73	58.82	61.26	83.22	86.23

Table 7. Base-level $F1$ scores (%) for the five domains.

The simple choice is to select the best base-level system for each domain. On the other hand, a more desirable approach is to try to exploit the diversity in the output of all systems, hoping to improve the best base-level results. Note that there is no generally accepted measure for diversity, but a variety of measures exist in the literature (Kuncheva and Whitaker, 2003). Ali and Pazzani (1996) define the similarity between two classifiers, as the conditional probability that both classifiers make an error, given that either of them makes an error.

Tables 8(a) to 8(e) are instances of the “contingency table” that was introduced by Freitag (2000) for measuring the similarity in the output of pairs of systems, and inspired by Ali and Pazzani’s measure. Each cell in a contingency table measures the conditional probability that the row system makes the correct prediction, given that the column system also predicts correctly. Tables 8(a) to 8(e) suggest that there is space for improving the best base-level system in each domain. For example, in CS courses we notice that only in 69% of the meta-level instances where HMMs yield a correct prediction, (LP)² also predicts correctly. Thus, in the remaining 31%, where HMMs predict correctly, (LP)² either predicts an incorrect field, or does not predict any field. Therefore, the performance of (LP)², which is the best system for this domain, can be further improved.

	BWI	HMM	(LP) ²
BWI	1	0.46	0.44
HMM	0.70	1	0.52
(LP) ²	0.89	0.69	1

(a) Courses

	BWI	HMM	(LP) ²
BWI	1	0.82	0.79
HMM	0.90	1	0.79
(LP) ²	0.69	0.62	1

(b) Projects

	BWI	HMM	(LP) ²
BWI	1	0.73	0.78
HMM	0.89	1	0.83
(LP) ²	0.87	0.76	1

(c) Laptops

	BWI	HMM	(LP) ²
BWI	1	0.83	0.86
HMM	0.91	1	0.85
(LP) ²	0.93	0.85	1

(d) Jobs

	BWI	HMM	(LP) ²
BWI	1	0.87	0.83
HMM	0.91	1	0.84
(LP) ²	0.95	0.92	1

(e) Seminars

Table 8. Contingency tables, measuring the agreement in the predictions of the base-level systems. Each cell is the probability that the row system makes a correct prediction, given that the column system makes a correct prediction.

6.2 Analysis of the Meta-level Instances

Each meta-level instance corresponds to a text fragment $t(s, e)$ that has been identified by at least one base-level system, together with the predicted fields for $t(s, e)$ by the base-level systems, the associated probabilistic estimates, and finally the correct human-annotated field for $t(s, e)$. Figure 4 shows a partition of the meta-level instances in the testing corpus, according to whether all systems agree on the same field for a fragment $t(s, e)$ or not.

The leftmost column for each domain in Figure 4 shows that there are regularities in the text documents that can be easily recognized by all available IE systems. For example, the fragment “TFT” is a typical instance of the field *screen type* in the domain of laptops that commonly appears in both training and testing corpus and thus easily detected by all systems.

Observing the rightmost column for each domain in Figure 4, leads to the interesting conclusion that situations where at least two base-level systems predict different fields for a text fragment are not frequent. In order to explain that, one should note that the IE systems exploit both the target content and the surrounding context of the relevant instances, and thus are capable of disambiguating among field instances with similar content. For example, instances of the fields *cdromSpeed* and *dvdSpeed* contain similar content, e.g. “24x”. A system that simply memorizes field instances verbatim predicts both fields for “24x”. Our base-level systems, on the other hand, are capable of examining surrounding tokens such as “cd” or “dvd”, and thus distinguish among the two fields. In some cases, however, those regularities in the context were difficult to find, either because they are less apparent, or due to limitations in the context that the base-level systems search for, thus resulting in contradictory fields.

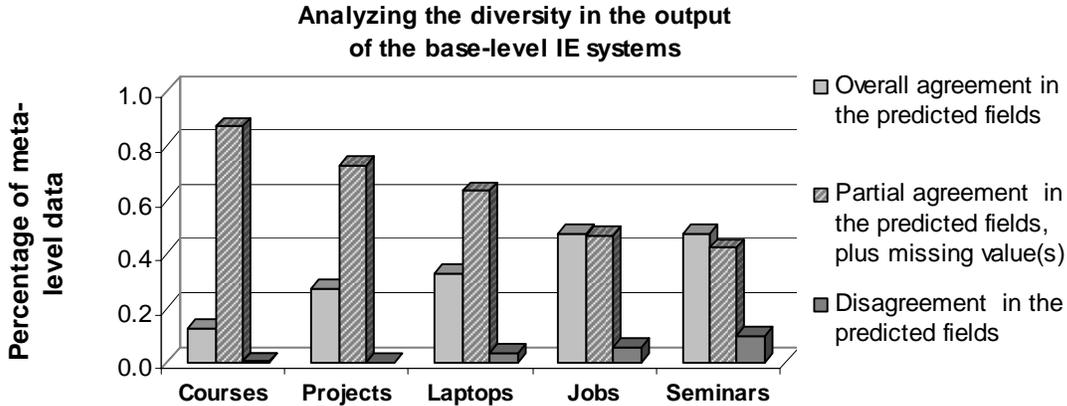


Figure 4. Partitioning the meta-level instances for each domain into three disjoint sets, according to whether all systems agree on the same field for a text fragment (left column), or some system(s) predict the same field while the other(s) abstain from prediction (middle column), or there are at least two contradictory predictions (right column).

For example, $(LP)^2$ explores a window of w tokens to the left and w tokens to the right of the starting and ending boundaries of the annotated field instances, where the value of w is predefined. Also in HMMs, a predefined number of *prefix* and *suffix* states model the immediate prefix and suffix respectively of the annotated field instances. The largest rate of disagreement appears in seminars (9.9%). This is because the ending time (field *etime*) of a seminar was many times confused with the starting one (field *stime*), thus increasing the size of the rightmost column in Figure 4 for this domain. In CS courses and research projects, contradictory predictions do not exceed 0.5% of all meta-level data, in laptops they are 3.7%, while for jobs they are 5.5%.

Since differences in the predicted fields for a text fragment are not frequent, the interesting question is what kind of disagreement can both voting and stacking exploit in pursuit of improved performance at meta-level? The answer lies in the middle column for each domain, which indicates that the majority of the meta-level instances derive from text fragments that have been assigned identical fields by some, but not all, system(s), while the remaining system(s) abstain from prediction. Since we deal with three systems, this corresponds to situations where either two systems predict the same field, plus a missing prediction by the third system or only one system predicts a field, plus two missing predictions. Therefore, we expect voting and stacking to exploit this kind of disagreement, leading to better results at meta-level. It is also interesting to observe the behaviour of stacking when the predictions by all systems are identical, according to the left column for each domain in Figure 4.

Note finally that the partition of meta-level data as shown in Figure 4, will allow for a more comprehensive comparison of stacking against voting, while also exploring the various aspects of the behaviour of stacking and voting, with respect to the varying degree of correlation in the output of the base-level systems. On the other hand, Table 8 shows a quantitative analysis of the disagreement among pairs of different IE systems that helps in determining whether there is space for improving the best system for each domain.

6.3 Results of Meta-level and Comparisons

Let $MVotM$ and $MVotF$ be the two majority voting settings, as defined in Section 3.2. The former setting ignores missing field prediction by some system, while the latter setting records missing prediction as “false”. Let also $PVotM$ and $PVotF$ be the two corresponding settings of voting using probabilities, as defined in Section 3.3. In $PVotM$, missing predictions are ignored. In $PVotF$, if the highest probability for a field f is less than 0.5, then f is rejected. Table 9 shows the best $F1$ scores obtained by all voting settings, stacking with nominal values, stacking with probabilities, and by the best base-level systems. Appendix A summarizes again all combination methods, along with a short description for each method.

	Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Courses	65.73	65.59	60.29	65.65	70.64	63.92	71.93
Projects	61.64	60.71	67.39	60.75	65.75	66.05	70.66
Laptops	63.81	62.37	67.60	62.76	71.03	68.46	71.55
Jobs	83.22	79.90	83.85	79.99	83.15	85.67	85.94
Seminars	86.23	86.87	87.13	86.90	88.02	88.48	90.03

Table 9. Best *F1* scores (%) obtained by all combination methods (voting and stacking) and by the best base-level system for each domain of interest. For fair comparisons, the results of a single classifier (*LogitBoost*) were used by both stacking settings.

Tables 10 to 12 compare all combination methods and the best base-level system, based on statistically significant *wins* against *losses*, in the five examined domains. Appendices B.2 to B.4 show detailed numerical values for the three measures of performance by all combination methods in each domain of interest. Detailed results per field can be found in (Sigletos, 2005).

	Best Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Best Base		5\0	0\5	5\0	2\2	0\5	0\5
MVotM	0\5		0\5	0\1	0\5	0\5	0\5
MVotF	5\0	5\0		5\0	5\0	2\0	3\1
PVotM	0\5	1\0	0\5		0\5	0\5	0\5
PVotF	2\2	5\0	0\5	5\0		0\5	0\5
Stacking Nominal	5\0	5\0	0\2	5\0	5\0		0\3
Stacking Probs	5\0	5\0	1\3	5\0	5\0	3\0	

Table 10. Statistically significant *wins* against *losses* in the five domains, based on *precision*, of the row system against the column one.

	Best Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Best Base		0\5	4\0	0\5	0\4	2\2	2\3
MVotM	5\0		5\0	0\1	5\0	5\0	5\0
MVotF	0\4	0\5		0\5	0\5	0\4	0\4
PVotM	5\0	1\0	5\0		5\0	5\0	5\0
PVotF	4\0	0\5	5\0	0\5		4\0	4\0
Stacking Nominal	2\2	0\5	4\0	0\5	0\4		0\3
Stacking Probs	3\2	0\5	4\0	0\5	0\4	3\0	

Table 11. Statistically significant *wins* against *losses* in the five domains, based on *recall*, of the row system against the column one.

	Best Base	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
Best Base		2\0	1\2	1\0	0\4	0\2	0\5
MVotM	0\2		1\3	0\1	0\5	0\3	0\5
MVotF	2\1	3\1		3\1	1\3	0\3	0\5
PVotM	0\1	1\0	1\3		0\5	0\3	0\5
PVotF	4\0	5\0	3\1	5\0		2\2	0\3
Stacking Nominal	2\0	3\0	3\0	3\0	2\2		0\4
Stacking Probs	5\0	5\0	5\0	5\0	3\0	4\0	

Table 12. Statistically significant *wins* against *losses* in the five domains, based on *F1*, of the row system against the column one.

6.4 Discussion

We observe that stacking with probabilities obtains a higher $F1$ score than the best base-level system for each domain. Precision is also improved (substantially for the domains of research projects and laptop products) in all five domains. Recall is improved in three domains but harmed in the remaining two. Stacking with simple nominal values, on the other hand, outperforms the best base-level $F1$ score in only two of the five domains. Note that the large improvement obtained by stacking with nominal values in projects and laptops is not consistent across all folds during evaluation, and thus measured as statistically insignificant. Overall, stacking with probabilities outperforms simple stacking with nominal values. Only for job offers, the obtained improvement in $F1$ against simple stacking was measured as statistically insignificant. Handling missing values in stacking did not significantly influence the results.

Regarding voting, $PVotF$ performs comparably or better than the best base-level system for each domain. Recall is improved by $PVotF$ at meta-level in most domains. Precision is however improved only in two domains (projects and laptops). Among all voting settings, $PVotF$ is best for courses, laptops and seminars, while $MVotF$ is slightly better only for jobs. For projects, the improvement obtained by $MVotF$ against $PVotF$ was measured as statistically insignificant. Overall, $PVotF$ is the best among all voting settings.

By comparing voting against stacking, we observe that stacking with probabilities outperforms $PVotF$ in all five domains, although the difference is statistically significant only in three domains. Unlike $PVotF$, stacking with probabilities is also consistently effective across all five domains, always outperforming the best base-level system for each domain. Moreover, stacking with probabilities always obtains more precise results than $PVotF$, at the cost of somewhat lower recall. Overall, stacking with probabilities achieves the best results among the combination methods that were evaluated.

6.4.1 Best Classifiers at Meta-level

An interesting result in the stacking with probabilities setting is that the highest $F1$ scores in all domains were obtained by the same classifier: *LogitBoost*. Only for projects, the classifier *j48* obtained a higher, but statistically insignificant, $F1$. On the other hand, *LogitBoost* using nominal values has not been consistently effective over all five domains. The best classifier using nominal values in CS courses was *IB1*, achieving however an insignificantly higher $F1$ score than the best base-level system for this domain.

Table 13 compares the six classifiers in stacking with probabilities, the setting that leads to the best meta-level results, based on statistically significant *wins* against *losses*, in the five examined domains. Appendix B.5 shows the $F1$ scores obtained for the five domains by all classifiers in both simple stacking with nominal values and stacking with probabilities.

	IB1	J48	LogitBoost	MLR	NaiveBayes	SMO
IB1		0\3	0\5	1\1	3\0	0\2
j48	3\0		0\3	2\0	5\0	1\0
LogitBoost	5\0	3\0		5\0	5\0	5\0
MLR	1\1	0\2	0\5		3\1	0\2
NaiveBayes	0\3	0\5	0\5	1\3		0\2
SMO	2\0	0\1	0\5	2\0	2\0	

Table 13. Stacking with probabilities. Statistically significant *wins* against *losses* in the five domains, based on $F1$, of the row classifier against the column one.

Clearly a variety of other classifiers could also be evaluated at meta-level. The aim of our experiments was to demonstrate the effectiveness of utilizing a common classifier in the context of combining multiple IE systems. Most of the employed classifiers have been also evaluated in recent studies for stacking, with *MLR* to be a state-of-the art approach (Ting and Witten, 1999; Seewald, 2003; Džeroski and Ženko, 2004). *MLR* did not prove to be that effective in our experiments for IE as in the cited studies for common classification. Nevertheless, all meta-level

classifiers in the cited studies, including *MLR*, have access to full probability distributions over all relevant classes, produced by the base-level classifiers. Such distributions are not typical for IE systems.

On the other hand, Table 13 shows that boosting simple decision stumps, through *LogitBoost*, was particularly effective at meta-level. Only for projects and seminars, the difference among *LogitBoost* and *j48* was measured as statistically insignificant.

6.4.2 Majority Voting

Tables 10 and 11 show that *MVoteM* improves recall but hurts precision, as compared to the best base-level system for each domain, while *MVoteF* improves precision but hurts recall. This contradicting behaviour is mainly due to situations where only one system predicts a field f for a text fragment, while the remaining two systems abstain. In such cases, if f is not the correct field for the fragment, this harms precision for *MVoteM*, which always accepts f . If f is the correct field, then this harms recall for *MVoteF*, which always returns “false”, which is the value with the highest count. Recall is harmed the most in CS courses by *MVoteF*, thus also harming $F1$, as it is shown in Appendix B.2, which is due to the fact that single predictions are mostly correct for this domain.

Note that since contradictory field predictions for a text fragment are not frequent, according to Figure 4, *MVoteM* does not actually behave like voting. In the overwhelming majority of meta-level data, only one field f participates in the vote counting process. The more systems predict f the higher the count is for f , which is then returned. However, if f is not the correct field, then there is no way to reject it, since missing predictions are ignored and thus precision is harmed. Nevertheless, the recall obtained by *MVoteM* approximates the maximum recall that can be obtained at meta-level for each domain, since each base-level system also contributes its uniquely identified correct instances. On the other hand, *MVoteF* does behave more like voting, since each missing prediction for a fragment is encoded with the value “false”, which then participates in the vote counting.

The overall conclusion is that neither majority voting setting has been consistently effective, based on $F1$, over all five domains. *MVoteF* achieves a higher $F1$ score at meta-level that is statistically significant only for projects and seminars, while *MVoteM* does not significantly improve the best base-level $F1$ score in any of the five domains. The large improvement that *MVoteF* obtains for laptops is not consistent over all folds during evaluation, and thus measured as statistically insignificant. In addition, we would like to *learn* when a field f is correct, instead of accepting f by *MVoteM*, if it has the highest count, or rejecting f by *MVoteF*, if the value with the highest count is “false”. Stacking using probabilities achieves this goal, outperforming both settings of majority voting in the five examined domains.

6.4.3 Voting Using Probabilities

Tables 9 to 12 show that the performance of *PVoteM* is similar to *MVoteM*. Since the overwhelming majority of meta-level instances contain no contradictory field predictions for a text fragment, according to Figure 4, the additional use of probabilities has not proved particularly useful for *PVoteM*. The higher the number of votes for f , the higher is the combined probabilistic estimate for the field. Only for laptops, the slight improvement in $F1$ by *PVoteM* is statistically significant. As a result, *PVoteM* approximates, slightly better than *MVoteM*, the maximum recall that can be obtained at meta-level for each domain.

Note that *PVoteF* performs an additional test to the field f that is returned by *PVoteM*, by examining whether or not the probability associated with f is greater than 0.5, and thus accepts or rejects f . This leads to more precise results for *PVoteF*, as compared to *PVoteM*, suffering a lower recall. The improvement in precision is however substantial and leads to higher $F1$ score for *PVoteF*, since most incorrect predictions are associated with a probability that is less than 0.5. Moreover, if the value with the highest count is “false”, i.e. two systems abstain from prediction, *PVoteF* examines the probability of the field f with the next highest count, i.e. the prediction of the remaining system. This explains both the higher recall and lower precision for *PVoteF*, against

MVoteF that always returns “false” in this case. Although *PVoteF* does significantly better than *MVoteF* in three domains, the decrease in precision by *PVoteF* is higher than the increase in recall for the remaining two (projects and jobs), thus resulting in a comparable or better *F1* score for *MVoteF*. This is due to more situations for the two domains where single predictions are both incorrect and assigned a high probability, and thus correctly rejected by *MVoteF*, while incorrectly accepted by *PVoteF*.

Although the calibrated probabilities of correctness are more meaningful and consistent than confidence scores, choosing 0.5 as a threshold below which to reject predictions may not always be an optimal choice, as showed in Figure 5 for CS courses.

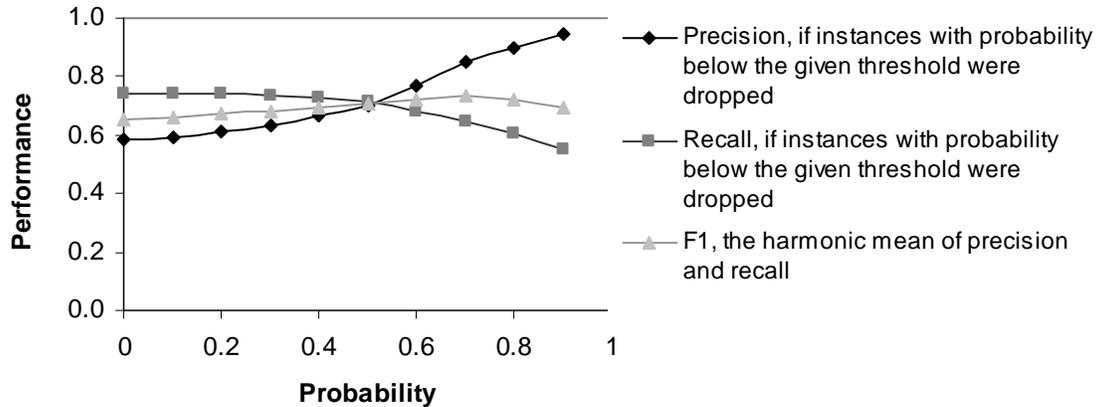


Figure 5. Precision, recall and *F1* by *PVoteF* for the domain of CS courses, regarding the threshold below which, the predicted instances are removed.

The performance of *PVoteF* for a probability threshold zero equals that of *PVoteM*, where no prediction is rejected. Increasing the threshold, below which predictions are rejected, a tradeoff between recall and precision is observed, reflecting the natural fact that a rejected prediction may be either a correct one, thus harming both precision and recall, or incorrect and thus improving precision. Nevertheless, incorrect predictions are mostly rejected as threshold increases, since most correct predictions are assigned a high probability.

As already mentioned in Section 2.3.3, the mapping of confidences to probabilities takes place by validating the performance of each base-level system on a hold-out set or by cross-validation. This only approximates the *true* probability distribution in the predictions of an IE system over the space of all possible relevant pages. As a result, the optimal threshold below which to reject predictions may vary for different collections of pages. In Figure 5, despite the fact that both precision and recall are almost equally balanced at threshold 0.5, the optimal *F1* score (73.5%) is achieved at threshold 0.7. For jobs, the optimal *F1* score (84.08%) is also achieved at threshold 0.7. All differences were measured as statistically significant. For research projects, seminars and laptops, the best *F1* score remains at threshold 0.5.

However, instead of optimizing the threshold empirically, e.g. in a jack-knifing procedure, it is much more interesting to try to *learn* whether or not to accept a prediction. In other words, we would like to learn the correlation among the probabilistic estimates, returned by the individual base-level systems, towards better meta-level results. Stacking using probabilities achieves this goal in most domains by outperforming *PVoteF*, while obtaining more precise results even when both settings perform comparably.

6.4.4 Multistrategy Learning for Information Extraction

Table 14 compares the *F1* scores obtained by the multistrategy learning setting, as described in Section 2.3.3, against the best obtained scores at the base-level, voting using probabilities, and the results presented in (Freitag, 2000). Results are only presented for the domains of CS courses, research projects and seminars, since these domains were used by Freitag (2000).

	Best Base	PVotM	PVotF	Multistrategy	Multistrategy (Freitag, 2000)
crsNumber	94.46	95.72	95.92	94.91	88.9
crsTitle	70.05	73.50	72.34	73.29	62.0
crsInst	48.21	50.81	57.76	50.81	49.8
projMember	65.00	63.16	68.94	63.09	45.5
projTitle	39.66	34.26	32.88	35.65	34.1
stime	99.09	99.51	99.51	99.42	99.3
etime	97.62	89.09	96.68	67.15	94.3
speaker	73.41	75.88	75.40	75.58	66.2
location	77.43	81.82	81.83	81.82	79.7

Table 14. Best per field *F1* scores (%) by base-level, voting and multistrategy learning for the domains of CS courses, research projects and seminar announcements.

Our results for multistrategy perform comparably or better for most fields against the ones in (Freitag, 2000), which is partially due to the higher performance of our systems. Multistrategy learning and *PVotM* behave similarly for all fields but for *etime*. Instances of *etime* are many times confused with instances of *stime* by our IE systems, thus resulting in contradictory field predictions. Multistrategy, however, considers each field in isolation and thus fails to distinguish among the two fields. Voting handles better field ambiguities, by selecting the one with the highest probability, thus leading in significantly higher *F1* score than multistrategy for ambiguous fields. In some cases the highest probability does not correspond to a correct prediction, which justifies why the *F1* for *etime* is not improved monotonically at meta-level.

Contradictory field predictions do not occur frequently in most fields, and thus *PVotM* mostly handles each field in isolation, as multistrategy does by default. In such cases, and just like *PVotM*, the success of multistrategy strongly depends on how the incorrect predictions by all base-level systems correlate. The more incorrect instances each system uniquely identifies, the higher the decrease in precision is, since missing predictions are ignored and there is no correct field to contradict the incorrect one during voting. This may lead to a worse *F1* score, compared to the best base-level system, as in research projects. Overall, *PVotF* is the best of the voting settings that were evaluated in this article.

7 Explaining the Results

In Section 6.2, we partitioned the meta-level instances according to how the base-level systems correlate in their output. In this section we compare stacking against voting with respect to this diversity analysis. The aim is to provide useful insight into the behaviour of voting and stacking by comparatively studying their performance, based on the varying degree of disagreement in the output of the base-level systems. In the interest of conducting a fair analysis, the results of a single classifier (*LogitBoost*) are used for stacking.

7.1 Analyzing Cases of Complete Agreement in the Output of the Base-level Systems

Figure 6 compares all combination methods, based on the number of correctly classified meta-level instances, when all base-level systems agree. These instances correspond to the left column in Figure 4 for each domain. Recall that each meta-level instance can be classified into one of the values $\{f^1 \dots f^Q, false\}$, where $\{f^1 \dots f^Q\}$ the relevant fields and *false* a special value indicating that the text fragment, which corresponds to the meta-level instance, is irrelevant, and thus none of the base-level systems should have predicted a field for it.

Stacking with probabilities performs slightly better than all other combination methods, while *PVotF* follows. In other words, stacking proved to be slightly more useful even when the predictions by the base-level systems are identical. Note that since all three systems agree on the same field, all voting settings except *PVotF* return the same value. Only *PVotF* has the additional option of rejecting a prediction by all systems, if the combined estimate is less than a given threshold, which has proven slightly useful only for projects and laptops.

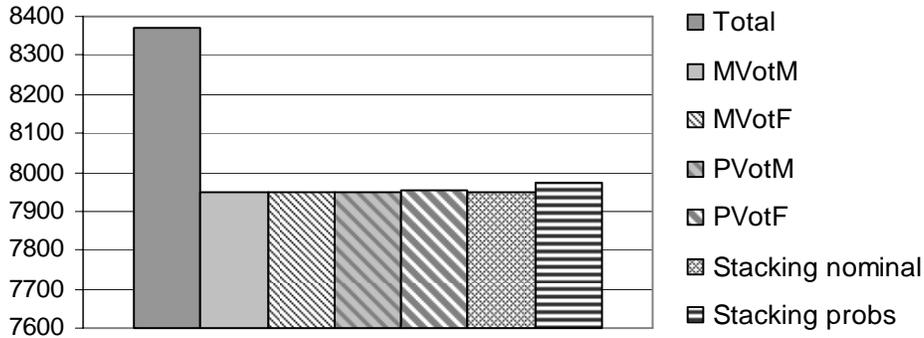


Figure 6. Comparing all combination methods, when all three base-level systems agree on the same field. Sum of correctly classified meta-level instances over all five domains.

Comparing the total number of meta-level instances where all three base-level systems agree (leftmost column in Figure 6) and the results obtained by all combination methods, we conclude that the returned field is not always correct. This was mostly observed in the Web domains and it is partly due to errors during annotation, and mainly due to captured regularities in the text that are not always reliable. For example, “TFT” may be the type of a separate screen product that is described in the same page with a laptop. Similarly for projects, some faculty or student names are former project members, and thus have not been annotated by the human expert. In such cases, stacking with probabilities learned to reject, i.e. classify as “false”, some of those erroneous predictions. We did not expect to do much better without encoding other features in the meta-level vectors.

The last observation indicates a limitation of our base-level systems which is not unknown in the literature. The limitation is that the IE systems parse a document as a linear sequence of tokens and thus ignore hierarchical structure available within an HTML or XML document, e.g. through using the document object model (DOM). Therefore, our IE systems only capture regularities in text that concern sequences of tokens within the target content and surrounding content of the relevant instances and thus fail to generalize over hierarchical information. For example, we would like to exploit HTML information to separate laptops from other products that are described within the same page.

There exist approaches in the literature that exploit HTML or XML structure in the context of IE, but they mostly suffer from the need of extensive manual effort and/or the lack of adaptability to different structure formats. For example, STALKER (Muslea et al., 2001) learns patterns that match certain sequences of tokens/wildcards, separated by irrelevant intermediate text. This however requires the manual construction and use of an “embedded-catalog” (EC) tree for Web pages that share the same structure in their content, and thus separate trees have to be manually constructed for different structure formats.

Davulcu et al. (2002) exploit DOM-based information in conjunction with a hand-crafted ontology for IE from Web pages with different structure. Incorporating combination methods for IE with their techniques is an issue to be investigated. For example, the hand-crafted patterns for the item-identifiers within the ontology could be replaced by more complex patterns induced by combining a set of IE systems, and thus reduce the effort required for engineering the ontology. Initial efforts towards incorporating machine learning into ontology engineering already exist (Valarakos et al., 2004).

7.2 Analyzing Cases of Partial Agreement in the Output of the Base-level Systems

Figure 7 compares all combination methods based on the number of correctly classified meta-level instances, when some base-level system(s) agree on the same field, while the remaining one(s) abstain from prediction. These instances correspond to the middle column for each domain in Figure 4. In order to analyze the results into greater depth, Figure 7 presents separately the

results, based on whether a single base-level system predicts a field, or exactly two out of the three systems agree on the same field.

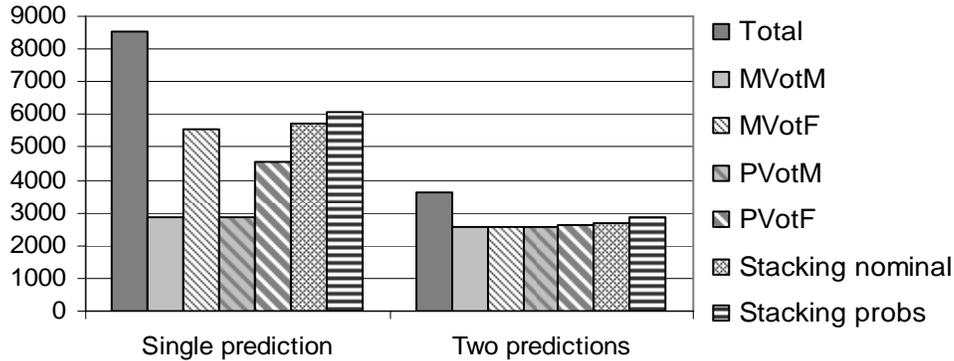


Figure 7. Comparing all combination methods, when either a single or exactly two base-level systems agree on the same field (set of columns on the left and right respectively). Sum of correctly classified meta-level instances over all five domains.

Figure 7 shows the superiority of stacking with probabilities in both situations. Handling missing values does not significantly influence the results. The left part in Figure 7 also confirms the complementary behaviour of *MVotM/PVotM* with *MVotF*. When only one system predicts a field for a text fragment, then the column size of *MVotM/PVotM* equals the number of cases where the predicted field is correct. The size of *MVotF* equals the number of cases where the predicted field is incorrect, and thus “false” is returned. The large size of the *MVotF* column in the left part of Figure 7 indicates that single-field predictions are more probably incorrect. The left part of Figure 7 shows that stacking with probabilities learns more than simple stacking with nominal values does and goes beyond what *MVotM/PVotM* and *MVotF* straightforwardly deduce in a contradicting manner, by obtaining a higher accuracy than all settings.

When two predictions agree on the same field and the third one is missing, all voting settings, except *PVotF*, obviously return the same field, as also shown in the right part of Figure 7. Stacking with nominal values and *PVotF* perform slightly better than the other voting settings. On the other hand, stacking with probabilities again learns something more than simple stacking does and goes beyond what all voting settings straightforwardly deduce.

7.3 Analyzing Cases of Disagreement in the Output of the Base-level Systems

Figure 8 compares all combination methods, based on the number of correctly classified meta-level instances, when at least two base-level systems contradict in their field predictions. These instances correspond to the right column in Figure 4 for each domain.

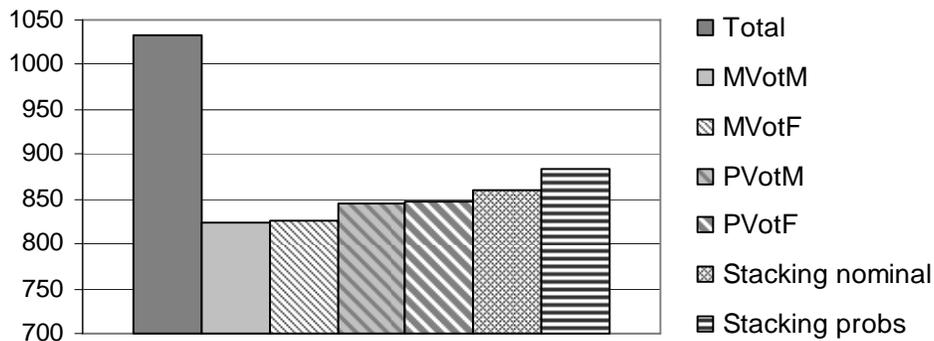


Figure 8. Comparing all combination methods when the base-level systems disagree. Sum of correctly classified meta-level instances over all five domains.

Figure 8 confirms the superiority of stacking with probabilities. Figures 6 to 8 indicate that there is enough room for further improving the results. On the other hand, the presented results are very positive, considering the exploitation at meta-level of simple nominal values and probabilities of correctness in the output of the base-level systems.

7.4 Comparing by Meta-level Classification Accuracy

Figures 6 to 8 compare voting and stacking using the number of correctly classified meta-level instances, summed over all domains, with respect to the different degree of agreement in the output of the base-level systems. *Classification accuracy* can then be defined as the fraction of meta-level instances that have been correctly classified. Table 15 compares again all combination methods, using classification accuracy as a measure of performance when estimating the statistically significant *wins* against *losses*.

	MVotM	MVotF	PVotM	PVotF	Stacking Nominal	Stacking Probs
MVotM		0\4	0\1	0\5	0\5	0\5
MVotF	4\0		4\0	2\0	0\3	0\5
PVotM	1\0	0\4		0\5	0\5	0\5
PVotF	5\0	0\2	5\0		0\3	0\5
Stacking Nominal	5\0	3\0	5\0	3\0		0\4
Stacking Probs	5\0	5\0	5\0	5\0	4\0	

Table 15. Statistically significant *wins* against *losses*, based on classification accuracy, in the five domains, of the row IE system against the column one.

Table 15 shows the clear superiority of stacking with probabilities over all voting settings. This is in contrast to Table 12, which shows that *PVotF* performs comparably to stacking with probabilities in two of the five examined domains. Moreover, *MVotF* seems to be the best voting setting, which also contradicts Table 12, where *PVotF* is the best setting. Those contradicting conclusions are due to the different metrics that are employed in the two tables. Table 12 shows statistically significant *wins* against *losses*, based on the micro-average *F1* that is measured over the relevant fields $\{f^1 \dots f^Q\}$ for each domain. On the other hand, the classification accuracy that is used in Table 15 is defined over all possible values $\{f^1 \dots f^Q, false\}$ that a meta-level instance can be classified into, including the *false* value. Actually, classification accuracy is identical to micro-average precision over all classes $\{f^1 \dots f^Q, false\}$ for a domain of interest.

Classification accuracy is typically used for comparing different classifiers over all class values. In IE, however, the class value *false* has a special interpretation, since no such field is annotated by the human expert. Similarly none of the employed systems at the base-level predict the value *false* for a text fragment. The value *false* is, however, an option for the meta-level classifier, indicating that at least one of the base-level systems has incorrectly predicted a field for an irrelevant text fragment. However, the evaluation takes place by comparing the relevant instances in the hand-filled templates and the corresponding templates filled by the IE systems, at either base or meta-level. Therefore, the evaluation metrics *precision*, *recall* and *F1*, are naturally defined over the relevant domain fields $\{f^1 \dots f^Q\}$ ignoring *false*.

7.5 Stacking Pairs of Information Extraction Systems

Experiments were also conducted on stacking and voting of pairs of base-level systems, trying to investigate whether combining all three systems provides added value over combining pairs of systems. Table 16 compares stacking with probabilities (again using the same classifier, *LogitBoost*, for fair comparisons) for all possible combinations of base-level systems. The comparison is based on statistically significant *wins* against *losses*, based on *F1*, in the five examined domains. Voting on pairs of base-level systems did not provide any statistically significant added value over stacking in the domains of interest. Therefore, the results of voting on pairs are omitted here.

	BWI +HMM	BWI+(LP) ²	HMM+(LP) ²	BWI+HMM+(LP) ²
BWI +HMM		0\4	0\5	0\5
BWI+(LP) ²	4\0		0\4	0\5
HMM+(LP) ²	5\0	4\0		0\4
BWI+HMM+(LP) ²	5\0	5\0	4\0	

Table 16. Statistically significant *wins* against *losses*, based on *F1*, in the five examined domains, of stacking with probabilities the row base-level systems, against stacking the column ones.

In one domain (CS courses), stacking HMMs with (LP)² results in a statistically insignificant difference in *F1* against stacking all three systems. This concludes that the contribution of BWI to the performance of stacking is not significant for this domain. Moreover, stacking HMMs with (LP)² proves better than stacking BWI with (LP)², which is not always justified by the performance of the individual base-level systems. For example, BWI obtains a higher *F1* than HMMs in jobs and seminars. Table 8 explains this behaviour, by observing a higher degree of correlation among (LP)², which is the best system for jobs and seminars, and BWI, than the corresponding one among (LP)² and HMMs.

The results of Appendix B.1 show that BWI suffers from lower recall in most domains, as compared to HMMs and (LP)². A general guideline for choosing which base-level systems to stack would be therefore to prefer systems biased towards recall, rather than precision, since stacking always obtains more precise results at meta-level. Higher recall suggests higher chance of covering instances that have not been identified by other systems, and thus leading to a higher degree of disagreement, in favour of stacking. For example, HMMs obtain higher recall, yet lower precision, than BWI in all five domains. Nevertheless, stacking HMMs with (LP)² is the best pair-wise combination scheme, according to Table 16. In Section 4.2, the choice of systems biased towards recall is also suggested, since higher recall leads to higher chance of minimizing the number of cases where relevant text fragments have not been identified by any base-level system and thus cannot be also identified at the meta-level.

The only exception to the above rule of thumb for high-recall base-level systems is in the domain of research projects, where (LP)² obtains a significantly lower recall than HMMs, but stacking BWI with (LP)² performs better than stacking BWI with HMMs. Table 8 explains again this behaviour, by observing a higher degree of correlation among HMMs and BWI, than the corresponding one among (LP)² and BWI.

8 Concluding Remarks

Though effective in improving the performance of multiple learning algorithms, typically voting and stacking restrict their applicability to common classification. This article extended the applicability of voting and stacking to information extraction (IE), and demonstrated their effectiveness using a variety of different algorithms and domains. The disagreement in the output of the IE systems that were employed at base-level has been successfully exploited by voting and stacking, leading to higher extraction performance at meta-level.

Experimental results have also shown that voting and stacking work best when using the confidence scores by the individual base-level systems that have been converted into probabilistic estimates of correctness. Voting using probabilities and setting a threshold, below which meta-level instances are rejected, proved particularly effective in most domains by outperforming the best base-level systems. Stacking using probabilities, on the other hand, proved consistently effective over all domains of interest, doing comparably or significantly better than voting. Precision was always improved by stacking at meta-level, as compared to the best base-level systems, while recall was improved in most domains. Whenever voting and stacking were doing comparably, stacking still obtained more precise results.

Since IE has been transformed into a common classification task at the meta-level, there are many opportunities for further improving the extraction performance. The experimental results that were presented for stacking in this article are encouraging, considering also the simplicity of

the features in the meta-level vectors that represent only the output of the base-level systems. Additional information could be exploited by stacking towards better results that further justify the additional computational cost over voting. In the domain of laptop products, for example, instances of “processor speed” appear typically after “processor name” instances, while instances of “ram” usually follow. Exploring such dependencies among extraction fields, or possibly other sources of information, could lead to useful extra features within the meta-level vectors to be exploited by the classifiers. The combination of different classifiers at a higher meta-level could also be examined.

A different stacking strategy could be applied by considering each field in isolation during combination, as proposed in (Freitag, 2000). In that case, a separate cross-validation process would take place in the base-level data set for each relevant field, and the problem would be transformed into a binary-learning task at the meta-level. Such a strategy would also deal with a limitation of cross-validation procedures over text documents that concerns *stratification*. In common classification over feature vectors, a similar distribution of classes is maintained in each fold. In IE, however, typically there is a different distribution of fields in each document and thus it is hard to approximate the same distribution of the fields in each fold. The penalty of stacking separately each field is that we cannot take advantage of the cases of contradictory field predictions in the output of the base-level systems.

Creating feature vectors is just one method of handling the meta-level data. Alternative methods can be investigated. An interesting extension is to appropriately encode the information available at the meta-level as special tags, which can be either embedded within the text or used as additional token features. This would allow the training of common IE systems also at the meta-level, since the meta-level data set would again consist of the same set of annotated text documents, including the additional meta-level information embodied within the text. This would also be aligned with Wolpert’s two major features of stacking: that data sets at both base-level and meta-level are of equal size, while a learning algorithm which is applied at the base-level can also be applied at the meta-level.

This article contributes to the direction of realizing the high potential of combination methods in the context of accurately identifying relevant information within the abundant of online text, aiming at a framework that can be easily adapted to new domains.

Acknowledgements

The first author was partially supported by a research scholarship from the Institute of Informatics and Telecommunications of NCSR “Demokritos”, Athens, Greece. The authors are grateful to Fabio Ciravegna for offering (LP)² and to Dayne Freitag for offering the two annotated WebKB data sets and seminar announcements. Many thanks to the editor and the anonymous reviewers for their useful comments that helped improving the article.

Appendix A: Summary of the Combination Methods for Information Extraction

Combination method	Short description	Described in section
MVotM	Majority voting. Missing values are ignored	3.2
MVotF	Majority voting. Missing values are encoded as special “false” values, indicating rejection of prediction	3.2
PVotM	Voting with probabilities. Missing values are ignored	3.3
PVotF	Voting with probabilities. A threshold is set (typically 0.5), for accepting/rejecting predictions	3.3
Stacking Nominal	Stacking using simple nominal values	4.2, 4.3
Stacking Probs	Stacking using probability values	4.4

Appendix B: Results of Base-level and Meta-level for the Five Domains of Interest

B.1 Results of Base-level

%	CS courses			Research projects			Laptop products		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BWI	74.55	39.10	51.30	60.05	61.47	60.75	74.99	53.23	62.26
HMM	60.50	58.29	59.39	56.24	68.18	61.64	62.29	65.42	63.81
(LP) ²	71.39	60.90	65.73	63.31	54.92	58.82	63.24	59.41	61.26

%	Job announcements			Seminar announcements		
	Precision	Recall	F1	Precision	Recall	F1
BWI	89.42	72.39	80.01	93.26	74.92	83.09
HMM	72.42	79.31	75.71	78.34	80.09	79.20
(LP) ²	87.70	79.18	83.22	91.39	81.63	86.23

B.2 Results of Majority Voting

%	Precision			Recall			F1		
	Best Base	MVotM	MVotF	Best Base	MVotM	MVotF	Best Base	MVotM	MVotF
Courses	71.39	58.68	82.05	60.90	74.35	47.65	65.73	65.59	60.29
Projects	56.24	49.17	68.88	68.18	79.32	65.96	61.64	60.71	67.39
Laptops	62.29	52.89	80.41	65.42	76.00	59.05	63.81	62.37	67.60
Jobs	87.70	71.29	93.06	79.18	90.88	76.31	83.22	79.90	83.85
Seminars	91.39	86.93	97.55	81.63	86.82	78.72	86.23	86.87	87.13

B.3 Results of Voting Using Probabilities

%	Precision			Recall			F1		
	Best Base	PVotM	PVotF	Best Base	PVotM	PVotF	Best Base	PVotM	PVotF
Courses	71.39	58.78	70.16	60.90	74.35	71.12	65.73	65.65	70.64
Projects	56.24	49.20	63.31	68.18	79.37	68.38	61.64	60.75	65.75
Laptops	62.29	53.21	72.86	65.42	76.47	69.30	63.81	62.76	71.03
Jobs	87.70	71.37	80.08	79.18	90.98	86.45	83.22	79.99	83.15
Seminars	91.39	86.99	90.69	81.63	86.82	85.50	86.23	86.90	88.02

B.4 Results of Stacking Using a Single Classifier

%	Precision			Recall			F1		
	Best Base	Stacking Nominal	Stacking Probs	Best Base	Stacking Nominal	Stacking Probs	Best Base	Stacking Nominal	Stacking Probs
Courses	71.39	81.32	79.03	60.90	52.66	66.01	65.73	63.92	71.93
Projects	56.24	68.84	78.21	68.18	63.59	64.45	61.64	66.05	70.66
Laptops	62.29	79.52	84.49	65.42	60.10	62.04	63.81	68.46	71.55
Jobs	87.70	89.89	90.27	79.18	81.82	82.00	83.22	85.67	85.94
Seminars	91.39	92.56	94.69	81.63	84.74	85.80	86.23	88.48	90.03

B.5 Results of All Classifiers in Stacking

Comparison of classifiers is based on *F1*. For readability, *LB*=LogitBoost, *NB*=NaiveBayes

%	Stacking with nominal values						Stacking with probabilities					
	<i>IB1</i>	<i>j48</i>	<i>LB</i>	<i>MLR</i>	<i>NB</i>	<i>SMO</i>	<i>IB1</i>	<i>j48</i>	<i>LB</i>	<i>MLR</i>	<i>NB</i>	<i>SMO</i>
Courses	66.03	50.79	63.92	54.62	60.11	56.76	70.23	70.24	71.93	70.38	65.16	70.41
Projects	61.18	55.36	66.05	60.89	60.99	55.98	66.77	71.41	70.66	62.17	65.63	62.65
Laptops	64.43	62.56	68.46	66.23	67.29	66.65	69.49	70.66	71.55	70.00	61.36	71.02
Jobs	80.58	83.93	85.67	84.61	81.77	84.52	83.88	85.22	85.94	84.95	77.23	84.75
Seminars	87.34	87.63	88.48	88.22	87.06	88.09	88.45	89.66	90.03	88.78	88.49	88.82

References

- Ali, K.M., Pazzani, M.J., Error reduction through learning multiple descriptions. *Machine Learning*, 24, 173-202, 1996.
- Breiman, L., Bagging Predictors, *Machine Learning*, 24(2), 123-140, 1996.
- Breiman, L., Stacked Regressions, *Machine Learning*, 24, 41-48, 1996a.
- Califf, M.E., Mooney, R.J., Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction, *Journal of Machine Learning Research (JMLR)*, 4, 177-210, 2003.
- Chan, P., An Extensive Meta-Learning Approach for Scalable and Accurate Inductive Learning, *PhD Thesis*, Columbia University, 1996.
- Chang, C.H., Lui, S.C., IEPAD: Information Extraction based on Pattern Discovery, *In Proceedings of the Tenth International WWW conference*, 509-516, New York, USA, 2001.
- Chawathe, S., Molina, H-C., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., Widom, J., The TSIMMIS Project: Integration of Heterogeneous Information Sources, *In Proceedings of the Tenth Meeting of Information Processing Society of Japan (IPSJ)*, 7-18, 1994.
- Ciravegna, F., Adaptive Information Extraction from Text by Rule Induction and Generalization, *In Proceedings of the Seventeenth IJCAI Conference*. Seattle, 1251-1256, 2001.
- Ciravegna, F., Lavelli, A., LearningPinochio: Adaptive Information Extraction for Real World Applications, *Natural Language Engineering*, 1(1), 1-21, 2003.
- Cohen, W., Hurst, M., Jensen, L.S., A Flexible Learning System for Wrapping Tables and Lists in HTML Documents, *In Proceedings of the Eleventh International WWW conference*, Hawaii, USA, 2002.
- Craven, M., DiPasquo, D., Freitag, D., McCallum, A.K., Mitchell, T., Nigam, K., Slattery, S., Learning to extract symbolic knowledge from the World Wide Web, *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 509-516, 1998.
- Crescenzi, V., Mecca, G., Merialdo, P., RoadRunner: Towards automatic data extraction from large Web sites, *In Proceedings of the Twenty-seventh International Conference on Very Large Data Bases (VLDB)*, 109-118, Rome, Italy, 2001.
- Davulcu, H., Mukherjee, S., Ramakrishnan, I.V., Extraction Techniques for Mining Services from Web Sources, *IEEE International Conference on Data Mining (ICDM)*, 601-604, 2002.
- Defense Advanced Research Projects Agency (DARPA), *Proceedings of the Sixth Message Understanding Conferences (MUC-6)*, Morgan Kaufmann, 1995.
- Defense Advanced Research Projects Agency (DARPA), *Proceedings of the Seventh Message Understanding Conferences (MUC-7)*, Morgan Kaufmann, 1996.
- Dietterich, T.G., Machine Learning research: Four current directions. *AI Magazine*, 18(4), 97-136, 1997.
- Dietterich, T.G., Approximate Statistical Tests for Comparing Supervised Machine Learning Algorithms, *Neural Computing*, 10(7), 1895-1924, 1998.
- Domingos, P., Unifying instance-based and rule-based induction, *Machine Learning*, 24(2), 141-168, 1996.
- Džeroski, S., Ženko, B., Is Combining Classifiers Better than Selecting the Best One? *Machine Learning*, 54(3), 255-273, 2004.
- Florian, R., Cucerzan, S., Schafer, C., Yarowsky, D., Combining Classifiers for Word Sense Disambiguation, *Natural Language Engineering*, 1(1), 1-14, 2002.

- Freitag, D., Machine Learning for Information Extraction in Informal Domains, *Machine Learning*, 39, 169-202, 2000.
- Freitag, D., Kushmerick, N., Boosted Wrapper Induction, *In Proceedings of the Sixteenth National conference on Artificial Intelligence (AAAI-99)*, 59-66, 1999.
- Freitag, D., McCallum, A.K., Information extraction with HMMs and shrinkage, *AAAI-99 workshop on machine learning for information extraction*, 1999.
- Freitag, D., McCallum, A.K., Information extraction with HMM structures learned by stochastic optimization, *In Proceedings of the Seventeenth National conference on Artificial Intelligence (AAAI-00)*, 584-589, 2000.
- Freund, Y., Schapire, R., Experiments with a new boosting algorithm, *In Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, 148-156, Bari, Italy, 1996.
- Friedman, J., Hastie, T., Tibshirani, R., Additive Logistic Regression: a Statistical View Of Boosting, *Technical Report*, Stanford University, 1999.
- Halteren, H., Zavrel J., Daelemans, W., Improving Accuracy in Word Class Tagging through Combination of Machine Learning Systems, *Computational Linguistics*, 27(2), 199-230, 2001.
- John, G.H., Langley, P., Estimating Continuous Distributions in Bayesian Classifiers, *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI)*, 338-345, Morgan Kaufmann, 1995.
- Kauchak, D., Smarr, J., Elkan, C., Sources of Success for Boosted Wrapper Induction, *Journal of Machine Learning Research (JMLR)*, 5, 499-527, 2004.
- Kuncheva, L.I., Whitaker, C.J., Measures of Diversity in Classifier Ensembles and their Relationship with the Ensemble Accuracy, *Machine Learning*, 51, 181-207, 2003.
- Kushmerick, N., Wrapper Induction for Information Extraction, *PhD Thesis*, University of Washington, 1997.
- Lafferty, J., McCallum, A.K., Pereira, F., Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *In Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, 282-289, Williamstown, MA, USA, 2001.
- McCallum, A.K., Freitag, D., Pereira, F., Maximum entropy markov models for information extraction and segmentation, *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, 591-598, Stanford University, CA, USA, 2000.
- Michalski, R., Tecuci, G., Machine learning: A multistrategy approach, *Morgan Kaufmann*, 1994.
- Muslea, I., Minton, S., Knoblock, C., Hierarchical Wrapper Induction for Semistructured Information Sources, *Journal Of Autonomous Agents and Multi-Agent Systems*, 4, 93-114, 2001.
- Platt, J., Fast Training of Support Vector Machines using Sequential Minimal Optimization, *Advances in Kernel Methods - Support Vector Learning*, 185-208, MIT Press, 1999.
- Quinlan, R.J., C4.5: Programs for Machine Learning, *Morgan Kaufmann*, 1993.
- Rabiner, L., A tutorial on hidden Markov models and selected applications in speech recognition, *In Proceedings of the IEEE 77-2*, 257-286, 1989.
- RISE, A repository of online information sources used in information extraction tasks, *URL: <http://www.isi.edu/info-agents/RISE>*, 1998.
- Sebastiani, F., Machine Learning for Automated Text Categorization, *ACM Computing Surveys (CSUR)*, 34 (1), 1-47, 2002.

- Seewald, A., Towards understanding stacking, *PhD Thesis*, Department of Informatics, Technical University of Wien, Austria, 2003.
- Seymore, K., McCallum, A.K., Rosenfeld, R., Learning hidden Markov model structure for Information Extraction, *Journal of Intelligent Information Systems (JIIS)*, 8(1), 5-28, 1999.
- Sigletos, G., Voting and stacking for information extraction: Extended results, *Technical Report DEMO 2005/3, NCSR Demokritos*, 2005.
- Sonderland, S., Learning Information Extraction Rules for Semi-structured and Free Text, *Machine Learning*, 34-(1/3), 233-272, 1999.
- Ting, K., Witten, M., Issues in stacked generalization, *Journal of Artificial Intelligence Research (JAIR)*, 10, 271-289, 1999.
- Thompson, C.A., Califf, M.E., Mooney, R.J., Active Learning for Natural Language Parsing and Information Extraction, *In Proceedings of the Sixteenth International Machine Learning Conference (ICML)*, 406-414, Bled, Slovenia, 1999.
- Valarakos, A., Paliouras, G., Karkaletsis, V., Vouros G., Enhancing Ontological Knowledge through Ontology Population and Enrichment, *In Proceedings of the Fourteenth International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, LNAI-3257, 144-156, Springer, 2004.
- Witten, I., Frank, E., Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, *Morgan Kaufmann*, 2000.
- Wolpert, D., Stacked Generalization, *Neural Networks*, 5(2), 241-260, 1992.

Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models

Neil Lawrence

NEIL@DCS.SHEF.AC.UK

*Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street
Sheffield, S1 4DP, U.K.*

Editor: Aapo Hyvärinen

Abstract

Summarising a high dimensional data set with a low dimensional embedding is a standard approach for exploring its structure. In this paper we provide an overview of some existing techniques for discovering such embeddings. We then introduce a novel probabilistic interpretation of principal component analysis (PCA) that we term dual probabilistic PCA (DPPCA). The DPPCA model has the additional advantage that the linear mappings from the embedded space can easily be non-linearised through Gaussian processes. We refer to this model as a Gaussian process latent variable model (GP-LVM). Through analysis of the GP-LVM objective function, we relate the model to popular spectral techniques such as kernel PCA and multidimensional scaling. We then review a practical algorithm for GP-LVMs in the context of large data sets and develop it to also handle discrete valued data and missing attributes. We demonstrate the model on a range of real-world and artificially generated data sets.

Keywords: Gaussian processes, latent variable models, principal component analysis, spectral methods, unsupervised learning, visualisation

1. Introduction

Machine learning is often split into three categories: supervised learning, where a data set is split into inputs and outputs; reinforcement learning, where typically a reward is associated with achieving a set goal, and unsupervised learning where the objective is to understand the structure of a data set. One approach to unsupervised learning is to represent the data, \mathbf{Y} , in some lower dimensional embedded space, \mathbf{X} . In a probabilistic model the variables associated with such a space are often known as latent variables. In this paper our focus will be on methods that represent the data in this latent (or embedded, we shall use the terms interchangeably) space.

Our approach is inspired by probabilistic latent variable models. It has roots in previously proposed approaches such as density networks (MacKay, 1995) where a multi-layer perceptron (MLP) is used to provide a mapping from the latent projections, \mathbf{X} , to the observed data, \mathbf{Y} . A prior distribution is placed over the latent-space and the latent-space's posterior distribution is approximated by sampling. Density networks made use of the MLP to perform the mapping, Bishop et al. (1996) replaced the MLP with a radial basis function (RBF) network with the aim of decreasing the training time for the model. This model evolved (Bishop et al., 1998) into the generative topographic mapping (GTM) where the latent-space was now sampled on a uniform grid, and importance sampling is reinterpreted as the fitting of a mixture model via the expectation-maximisation (EM) algorithm.

This allows the points in the latent-space to be laid out on a uniform grid¹ (rather than sampled). This grid layout is shared with the self organising map (SOM) (Kohonen, 1990) and in Bishop et al. (1997) it was argued that the GTM provides a principled alternative to the self organising map.

The models outlined above are typically designed to embed a data set in two dimensions, they rely on either importance sampling, or a grid of points in the latent-space to achieve this embedding, this causes problems when the dimensionality of the latent-space increases. Point representations of the latent-space are useful because they allow for non-linear models: each point is easy to propagate through the non-linear mapping to the data-space. These non-linear mappings are designed to address the weaknesses in visualising data sets that arise when using standard statistical tools that rely on linear mappings, such as principal component analysis (PCA) and factor analysis (FA): with a linear mapping it may not be possible to reflect the structure of the data through a low dimensional embedding.

Principal component analysis seeks a lower dimensional sub-space (typically represented by its orthonormal basis) in which the projected variance of the data is maximised. If a two dimensional sub-space is sought then the projections may be visualised; but it may be necessary to include more latent dimensions to capture the variability (and therefore hopefully, but by no means necessarily the structure) in the data. Principal component analysis also has a latent variable model representation (Tipping and Bishop, 1999) which is strongly related to Factor Analysis (FA) (Bartholomew, 1987; Basilevsky, 1994). Both are linear-Gaussian latent variable models, but FA allows for a richer noise model than PCA (for recent work on non-linear factor analysis see Honkela and Valpola, 2005).

Naturally statisticians have not constrained themselves to linear methods when visualising data and in the next section we shall briefly review multidimensional scaling and related techniques that rely on *proximity data*.

1.1 Multidimensional Scaling and Kernel PCA

We have already mentioned several visualisation techniques which rely on learning a mapping from a latent-space (the embedded space) to the data-space. In this section we will briefly review methods that use *proximity data* to obtain a visualisation or embedding. Broadly speaking these methods are all variants or enhancements of the technique known as multidimensional scaling (MDS). In these methods, rather than observing data directly, information about the data set is summarised in an $N \times N$ matrix of either similarities or dissimilarities. Examples include distance matrices (a dissimilarity matrix) and kernel matrices (a similarity matrix). Each method we review provides answers to at least one of two questions.

1. How is the proximity matrix compiled?
2. How is the embedding developed from the proximity matrix?

Most of the variants of multidimensional scaling (Mardia et al., 1979) appear to focus on the second question. In classical MDS (Torgerson, 1952) an eigendecomposition of the centred similarity matrix² is performed. This is sometimes viewed as minimising a particular *stress function* where distances in the visualised space are matched to those in the data space. Attempting to preserve these distances is known as *metric MDS*, in *non-metric MDS* only the ordering of distances is preserved.

-
1. When sampling techniques are used the latent points will be in random positions.
 2. When the data is presented in the form of a distance or dissimilarity matrix a simple conversion may be performed to obtain a similarity matrix.

There are strong connections between MDS and kernel PCA (Schölkopf et al., 1998), some of which are formalised in Williams (2001). Kernel PCA also provides an answer to the first question—the suggestion is that the proximity data is provided by a positive semi-definite Mercer kernel that is computed on the data, \mathbf{Y} . The use of this kernel implies the existence of a non-linear mapping³ from the data-space to the latent-space (recall that the GTM and density networks perform the non-linear mapping in the opposite direction). The existence of this function is important as it allows data points which were not in the training set to be mapped to a position in the latent space without re-solving the eigenvalue problem. However, for both kernel PCA and MDS methods, it is not obvious how to project back from the latent-space to the data-space (this is known as the pre-image problem). Neither is it clear how to handle missing data⁴ as the proximity data matrix cannot normally be computed consistently if a particular attribute is not available.

Sammon mappings (Sammon, 1969) also attempt to match the embedded distances between points with the distances in the observed space (therefore they are a form of MDS). They suffer from the same weakness as MDS in that projection of data points which were not in the original data set can be computationally demanding, *i.e.* despite their name they do not provide an explicit mapping between the data and latent-space. The lack of a mapping was addressed by the Neuroscale algorithm of Lowe and Tipping (1996) a version of which was also suggested for MDS (Tipping, 1996).

Other recent work of importance which has focussed on forming the proximity matrix includes Isomap (Tenenbaum et al., 2000), where an approximation to geodesic distance is used and spectral clustering (see *e.g.* Shi and Malik, 2000) where the proximity data is derived from a graph.

In Table 1 we have summarised some of the properties of these algorithms/models. We have also included the model that is the subject of this paper, the Gaussian process latent variable model (GP-LVM).

In the remainder of this paper we will introduce the GP-LVM from the latent variable model perspective. The GP-LVM belongs to the same class of methods as density networks and the GTM, however there are also connections to classical MDS and kernel PCA. In particular, in the next section, we show that the approaches share an objective function. In Section 3 we will cover some of the algorithmic issues that arise with the model. The framework within which our GP-LVM is developed makes it straightforward to modify the approach for data for which a Gaussian noise model is not appropriate (such as binary or ordinal), this is discussed in Section 5. Handling of missing data attributes is also straightforward (Section 6). The algorithm’s characteristics are explored empirically in Section 7.

2. Gaussian Process Latent Variable Models

In this paper we present the Gaussian process latent variable model. As we shall see, the model is strongly related to many of the approaches that we have outlined above. There is a point representation in the latent-space (as there was for the GTM and density networks) and we will minimise an objective function that can be related to classical MDS and kernel PCA (see Section 2.6). Our starting point, however, will be a novel probabilistic interpretation of principal component analysis

3. A good reference which introduces Mercer kernels is Schölkopf and Smola (2001) Chapter 2.

4. Here, by missing data, we mean missing attributes which would normally be used in computing the proximity data matrix. For proximity data methods missing data can also mean elements missing from the proximity matrix, we do not discuss this case.

	Proximity	$\mathbf{X} \rightarrow \mathbf{Y}$	$\mathbf{Y} \rightarrow \mathbf{X}$	Non-linear	Probabilistic	Convex
PCA	I	Y	Y		I	Y
FA		Y	Y		Y	Y
Kernel PCA	Y		Y	Y		Y
MDS	Y			Y		
Sammon mapping	Y			Y		
Neuroscale	Y		Y	Y		
Spectral clustering	Y			Y		Y
Density Networks		Y		Y	Y	
GTM		Y		Y	Y	
GP-LVM	I	Y		Y	Y	

Table 1: Overview of the relationship between algorithms. A ‘Y’ indicates the algorithm exhibits that property, an ‘I’ indicates that there is an interpretation of the algorithm that exhibits the associated property. The characteristics of the algorithm are: *proximity*: is the method based on proximity data? $\mathbf{X} \rightarrow \mathbf{Y}$: does the method lead to a mapping from the embedded to the data-space? $\mathbf{Y} \rightarrow \mathbf{X}$: does the method lead to a mapping from data to embedded space? *Non-linear*: does the method allow for non-linear embeddings? *Probabilistic*: is the method probabilistic? *Convex*: algorithms that are considered convex have a unique solution, for the others local optima can occur.

which we will refer to as dual probabilistic principal component analysis (DPPCA). Dual probabilistic principal component analysis turns out to be a special case of the more general class of models we refer to as GP-LVMs.

2.1 Latent Variable Models

Typically we specify a latent variable model relating a set of latent variables, $\mathbf{X} \in \mathfrak{R}^{N \times q}$, to a set of observed variables, $\mathbf{Y} \in \mathfrak{R}^{N \times D}$, through a set of parameters. The model is defined probabilistically, the latent variables are then marginalised and the parameters are found through maximising the likelihood.

Here we consider an alternative approach: rather than marginalising the latent variables and optimising the parameters we marginalise the parameters and optimise the latent variables. We will show how the two approaches can be equivalent: for a particular choice of Gaussian likelihood and prior both approaches lead to a probabilistic formulation of principal component analysis (PCA). In the next section we will review the standard derivation of probabilistic PCA (Tipping and Bishop, 1999), then we will show how an alternative probabilistic formulation may be arrived at (see also Appendix A).

2.2 Probabilistic PCA

Probabilistic PCA (PPCA) is a latent variable model in which the maximum likelihood solution for the parameters is found through solving an eigenvalue problem on the data’s covariance matrix

(Tipping and Bishop, 1999). Let's assume that we are given a set of centred D -dimensional data $\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_N]^T$. We denote the q -dimensional latent variable associated with each data point by \mathbf{x}_n . The relationship between the latent variable and the data point is linear with noise added,

$$\mathbf{y}_n = \mathbf{W}\mathbf{x}_n + \eta_n$$

where the matrix $\mathbf{W} \in \mathcal{R}^{D \times q}$ specifies the linear relationship between the latent-space and the data space and the noise values, $\eta_n \in \mathcal{R}^{D \times 1}$, are taken to be an independent sample from a spherical Gaussian distribution⁵ with mean zero and covariance $\beta^{-1}\mathbf{I}$,

$$p(\eta_n) = N(\eta_n | \mathbf{0}, \beta^{-1}\mathbf{I}).$$

The likelihood for a data point can then be written as

$$p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}, \beta) = N(\mathbf{y}_n | \mathbf{W}\mathbf{x}_n, \beta^{-1}\mathbf{I}). \quad (1)$$

To obtain the marginal likelihood we integrate over the latent variables,

$$p(\mathbf{y}_n | \mathbf{W}, \beta) = \int p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}, \beta) p(\mathbf{x}_n) d\mathbf{x}_n, \quad (2)$$

which requires us to specify a prior distribution over \mathbf{x}_n . For probabilistic PCA the appropriate prior is a unit covariance, zero mean Gaussian distribution,

$$p(\mathbf{x}_n) = N(\mathbf{x}_n | \mathbf{0}, \mathbf{I}).$$

The marginal likelihood for each data point can then be found analytically (through the marginalisation in (2)) as

$$p(\mathbf{y}_n | \mathbf{W}, \beta) = N(\mathbf{y}_n | \mathbf{0}, \mathbf{W}\mathbf{W}^T + \beta^{-1}\mathbf{I}).$$

Taking advantage of the independence of the data points, the likelihood of the full data set is given by

$$p(\mathbf{Y} | \mathbf{W}, \beta) = \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{W}, \beta). \quad (3)$$

The parameters \mathbf{W} can then be found through maximisation of (3). Tipping and Bishop (1999) showed that there is an analytic solution to this maximisation. This solution is achieved when the matrix \mathbf{W} spans the principal sub-space of the data. This model therefore has an interpretation as a *probabilistic* version of PCA.

Marginalising the latent variables and optimising the parameters via maximum likelihood is a standard approach for fitting latent variable models. In the next section we will introduce an alternative approach. Instead of optimising parameters and marginalising latent variables we will suggest the dual approach of marginalising parameters, \mathbf{W} , and optimising with respect to latent variables, \mathbf{X} . For a particular choice of prior distribution on \mathbf{W} this probabilistic model will also turn out to be equivalent to PCA.

5. We use the notation $N(\mathbf{z} | \mu, \Sigma)$ to denote a Gaussian distribution over \mathbf{z} with mean μ and covariance Σ .

2.3 Probabilistic PCA through Latent Variable Optimisation

In the Bayesian framework parameters, such as \mathbf{W} , are viewed as random variables. The Bayesian methodology requires a suitable choice of prior for \mathbf{W} , and then proceeds to treat the parameters as latent variables. A simple choice of prior that is conjugate to (1) would be a spherical Gaussian distribution:

$$p(\mathbf{W}) = \prod_{i=1}^D N(\mathbf{w}_i | \mathbf{0}, \mathbf{I})$$

where \mathbf{w}_i is the i th row of the matrix \mathbf{W} . Unfortunately⁶ marginalisation of both \mathbf{W} and $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]^T$ is intractable. If we wish to proceed without turning to approximate methods we are faced with a choice over what to marginalise. The natural choice seems to be to marginalise $\mathbf{X} \in \mathfrak{R}^{N \times q}$ as typically it will be of larger dimension⁷ than $\mathbf{W} \in \mathfrak{R}^{D \times q}$. In practice though, it turns out that the two approaches are equivalent.

Marginalisation of \mathbf{W} is straightforward due to our choice of a conjugate prior. The resulting marginalised likelihood takes the form

$$p(\mathbf{Y} | \mathbf{X}, \beta) = \prod_{d=1}^D p(\mathbf{y}_{:,d} | \mathbf{X}, \beta), \quad (4)$$

where we use $\mathbf{y}_{:,d}$ to represent the d th column of \mathbf{Y} and

$$p(\mathbf{y}_{:,d} | \mathbf{X}, \beta) = N(\mathbf{y}_{:,d} | \mathbf{0}, \mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}). \quad (5)$$

We now look to optimise with respect to the latent variables. As might be expected from the duality of (3) and (4), this optimisation is very similar to that presented in Tipping and Bishop (1999). Our objective function is the log-likelihood,

$$L = -\frac{DN}{2} \ln 2\pi - \frac{D}{2} \ln |\mathbf{K}| - \frac{1}{2} \text{tr}(\mathbf{K}^{-1} \mathbf{Y}\mathbf{Y}^T), \quad (6)$$

where

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}.$$

The gradients of (6) with respect to \mathbf{X} may be found (Magnus and Neudecker, 1999) as,

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{K}^{-1} \mathbf{Y}\mathbf{Y}^T \mathbf{K}^{-1} \mathbf{X} - D\mathbf{K}^{-1} \mathbf{X},$$

a fixed point where the gradients are zero is then given by

$$\frac{1}{D} \mathbf{Y}\mathbf{Y}^T \mathbf{K}^{-1} \mathbf{X} = \mathbf{X}.$$

In Appendix B we show how the values for \mathbf{X} which maximise the likelihood are given by

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T$$

6. If it were possible to marginalise both the parameters and latent variables analytically we could use Bayes factors to perform model selection (see, for example, Bishop, 1999).

7. The matrix \mathbf{X} will be of larger dimension than \mathbf{W} unless $D > N$, *i.e.* there are more features than data points.

where \mathbf{U} is an $N \times q$ matrix whose columns are the first q eigenvectors of $\mathbf{Y}\mathbf{Y}^T$, \mathbf{L} is a $q \times q$ diagonal matrix whose j th element is $l_j = \left(\lambda_j - \frac{1}{\beta}\right)^{-\frac{1}{2}}$ where λ_j is the eigenvalue associated with the j th eigenvector of $D^{-1}\mathbf{Y}\mathbf{Y}^T$ and \mathbf{V} is an arbitrary $q \times q$ rotation matrix. Here, and in what follows, we will assume that these eigenvalues are ordered according to magnitude with the largest being placed first. Note that the eigenvalue problem we have developed can easily be shown to be equivalent to that solved in PCA (see Appendix C), indeed the formulation of PCA in this manner is a key step in the development of kernel PCA (Schölkopf et al., 1998) where the matrix of inner products $\mathbf{Y}\mathbf{Y}^T$ is replaced with a kernel (see Tipping (2001) for a concise overview of this derivation). Our probabilistic PCA model shares an underlying structure with that of Tipping and Bishop (1999) but differs in that where they optimise we marginalise and where they marginalise we optimise.

2.4 Gaussian Processes

Gaussian processes (O’Hagan, 1992; Williams, 1998) are a class of probabilistic models which specify distributions over function spaces. While a function is an infinite dimensional object, a distribution over the function space can be considered by focussing only on points where the function is instantiated. In Gaussian processes the distribution over these instantiations is taken to be Gaussian. Modelling with Gaussian processes consists of first specifying a Gaussian process prior. Usually a Gaussian distribution is parameterised by a mean and a covariance. In the case of Gaussian processes the mean and covariance must be functions of the space on which the process operates. Typically the mean function is taken to be zero, while the covariance function is necessarily constrained to produce positive definite matrices.⁸

Consider a simple Gaussian process prior over the space of functions that are fundamentally linear, but are corrupted by Gaussian noise of variance $\beta^{-1}\mathbf{I}$. The covariance function, or kernel, for such a prior is given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j + \beta^{-1} \delta_{ij}, \quad (7)$$

where \mathbf{x}_i and \mathbf{x}_j are vectors from the space of inputs to the function and δ_{ij} is the Kronecker delta. If these inputs were taken from our embedding matrix, \mathbf{X} , and the covariance function was evaluated at each of the N points we would recover a covariance matrix of the form

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}, \quad (8)$$

where the element at the i th row and j th column of \mathbf{K} is given by (7). This is recognised as the covariance associated with each factor of the marginal likelihood for dual probabilistic PCA (5). The marginal likelihood for dual probabilistic PCA is therefore a product of D independent Gaussian processes. In principal component analysis we are optimising the parameters *and* input positions of a Gaussian process prior distribution where the (linear) covariance function for each dimension is given by \mathbf{K} .

2.5 Gaussian Process Latent Variable Models

The dual interpretation of probabilistic PCA described above points to a new class of models which consist of Gaussian process mappings from a latent space, \mathbf{X} , to an observed data-space, \mathbf{Y} . Dual

8. The positive definite constraint implies that these covariance functions are also valid Mercer kernels. It is therefore common to refer to the covariance function as a kernel. In this paper we shall use the two terms interchangeably.

probabilistic PCA is the special case where the output dimensions are *a priori* assumed to be linear, independent and identically distributed. However, each of these assumptions can be infringed to obtain *new* probabilistic models. Independence can be broken, for example, by allowing an arbitrary rotation on the data matrix \mathbf{Y} , the ‘identically distributed’ assumption can be broken by allowing different covariance functions for each output dimension.⁹ In this paper we focus on the third assumption, linearity. By replacing the inner product kernel with a covariance function that allows for non-linear functions we can obtain a non-linear latent variable model. Due to the close relationship with the linear model, which has an interpretation as probabilistic PCA, such a model can be interpreted as a non-linear probabilistic version of PCA.

2.6 Proximity Data and the GP-LVM

We indicated in the introduction that the GP-LVM has connections with proximity data based methods such as kernel PCA and classical MDS. These connections are through a unifying objective function which embraces all three models. In the next section we briefly introduce this objective function.

2.6.1 A UNIFYING OBJECTIVE FUNCTION

Classical MDS and kernel PCA rely on proximity data, such as similarity matrices. Let’s denote the matrix of similarities for these methods by \mathbf{S} . For the case of positive definite¹⁰ similarity measures the matrix \mathbf{S} can be interpreted as a covariance (or covariance function). The cross entropy between this Gaussian and the Gaussian process whose marginal likelihood was given in (4) is

$$-\int N(\mathbf{z}|\mathbf{0}, \mathbf{S}) \ln N(\mathbf{z}|\mathbf{0}, \mathbf{K}) d\mathbf{z} = \frac{N}{2} \ln 2\pi + \frac{1}{2} \ln |\mathbf{K}| + \frac{1}{2} \text{tr}(\mathbf{K}^{-1}\mathbf{S}). \quad (9)$$

If we substitute $\mathbf{S} = D^{-1}\mathbf{Y}\mathbf{Y}^T$ we see, up to a scaling of $-D$, that (9) becomes identical to (6). Taking $\mathbf{K} = \mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}$ and minimising (9) with respect to \mathbf{X} leads to a solution (Appendix B) of the form

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T.$$

Where the matrix $\mathbf{U} \in \mathbb{R}^{N \times q}$ has columns which are the eigenvectors of \mathbf{S} . For the specific case¹¹ where $\mathbf{S} = D^{-1}\mathbf{Y}\mathbf{Y}^T$ the optimisation is identical to that of dual probabilistic PCA. However in the more general case where \mathbf{S} is either a kernel function or simply a positive definite matrix of similarities kernel PCA and classical MDS are recovered. We also note that the entropy of $N(\mathbf{z}|\mathbf{0}, \mathbf{S})$ is constant in \mathbf{X} , we therefore may subtract it from our objective function without affecting the optimisation with respect to \mathbf{X} . The resulting objective function is then the Kullback-Leibler divergence (Kullback and Leibler, 1951) between the two Gaussians,

$$\begin{aligned} \text{KL}(N(\mathbf{z}|\mathbf{0}, \mathbf{S}) || N(\mathbf{z}|\mathbf{0}, \mathbf{K})) &= -\int N(\mathbf{z}|\mathbf{0}, \mathbf{S}) \ln \frac{N(\mathbf{z}|\mathbf{0}, \mathbf{K})}{N(\mathbf{z}|\mathbf{0}, \mathbf{S})} d\mathbf{z} \\ &= \frac{1}{2} \ln |\mathbf{K}| - \frac{1}{2} \ln |\mathbf{S}| + \frac{1}{2} \text{tr}(\mathbf{S}\mathbf{K}^{-1}) - \frac{N}{2}. \end{aligned}$$

9. A very simple example of this idea would be to allow different noise distributions on each output direction. The probabilistic model underlying factor analysis allows this flexibility (see, for example, Tipping and Bishop 1999).

10. The analysis that follows can be extended to positive semi-definite \mathbf{S} by adding a diagonal term, $\sigma^2\mathbf{I}$ to \mathbf{S} and considering the limit as $\sigma^2 \rightarrow 0$.

11. In the MDS literature this is also sometimes referred to as principal co-ordinate analysis.

With appropriate choice of \mathbf{S} and \mathbf{K} this is a valid objective function for PCA, kernel PCA, classical MDS and the GP-LVM. For kernel PCA \mathbf{S} is a ‘non-linear kernel’ and \mathbf{K} is the ‘linear kernel’. For the GP-LVM \mathbf{S} is the ‘linear kernel’ whereas \mathbf{K} is a ‘non-linear kernel’. In practice this means that the GP-LVM is harder to optimise (solving an eigenvalue problem is no longer sufficient) but the GP-LVM maintains a probabilistic interpretation that kernel PCA doesn’t have.

The methods overlap when both \mathbf{K} and \mathbf{S} are based on inner product matrices (as outlined for DPPCA above).

Note that when the similarity measure, \mathbf{S} , is not of the form of the inner product kernel the objective function no longer has an interpretation as a likelihood. Therefore our approach is *not* a probabilistic interpretation of multidimensional scaling: we refer the reader to MacKay and Zinnes (1986) and Oh and Raftery (2001) for details of probabilistic MDS methods.

2.6.2 A NOTE ON REVERSING THE KULLBACK-LEIBLER DIVERGENCE

The Kullback-Leibler divergence is an asymmetric measure of distribution divergence so it is natural to consider the effect of reversing the role of the distributions and taking expectations under the distribution governed by \mathbf{K} rather than that governed by \mathbf{S} . For this special case, the reversed KL divergence is very similar to the original, only all matrices \mathbf{K} and \mathbf{S} are now replaced with their inverses. So the new objective function is

$$L = \frac{1}{2} \ln |\mathbf{S}| - \frac{1}{2} \ln |\mathbf{K}| + \frac{1}{2} \text{tr}(\mathbf{K}\mathbf{S}^{-1}) - \frac{N}{2},$$

The minimum can again be found through an eigenvalue problem, but now the retained eigenvalues from \mathbf{K} are the smallest, rather than the largest. In this respect the model leads to *minor component analysis*.

3. Fitting a Non-linear GP-LVM

We saw in the previous section how PCA can be interpreted as a Gaussian process that maps latent-space points to points in data-space. The positions of the points in the latent-space can be determined by maximising the process likelihood with respect to \mathbf{X} . It is natural, therefore, to consider alternative GP-LVMs by introducing covariance functions which allow for non-linear processes. The resulting models will not, in general, be optimisable through an eigenvalue problem.

3.1 Optimisation of the Non-linear Model

In the previous section we saw for the linear kernel that a closed form solution could be obtained up to an arbitrary rotation matrix. Typically, for non-linear kernels, there will be no such closed form solution and there are likely to be multiple local optima. There is a wide choice of non-linear covariance functions, some of which will be reviewed in Section 7.1. To use a particular kernel in the GP-LVM we first note that gradients of (6) with respect to the latent points can be found through first taking the gradient with respect to the kernel,

$$\frac{\partial L}{\partial \mathbf{K}} = \mathbf{K}^{-1} \mathbf{Y} \mathbf{Y}^T \mathbf{K}^{-1} - D \mathbf{K}^{-1}, \tag{10}$$

and then combining it with $\frac{\partial \mathbf{K}}{\partial x_{n,j}}$ through the chain rule. As computation of (10) is straightforward and independent of the kernel choice we only require that the gradient of the kernel with respect to

the latent points can be computed. These gradients may then be used in combination with (6) in a non-linear optimiser to obtain a latent variable representation of the data. Furthermore, gradients with respect to the parameters of the kernel matrix may be computed and used to jointly optimise \mathbf{X} and the kernel's parameters.

The log-likelihood is a highly non-linear function of the embeddings and the parameters. We are therefore forced to turn to gradient based optimisation of the objective function. Scaled conjugate gradient (Møller, 1993) is an approach to optimisation which implicitly considers second order information while using a scale parameter to regulate the positive definitiveness of the Hessian at each point. We made use of scaled conjugate gradient (SCG) for our experiments.

3.2 Illustration of GP-LVM via SCG

To illustrate a simple Gaussian process latent variable model we turn to the ‘multi-phase oil flow’ data (Bishop and James, 1993). This is a twelve dimensional data set containing data of three known classes corresponding to the phase of flow in an oil pipeline: stratified, annular and homogeneous. In Bishop et al. (1998), see also Section 7.2.1, this data was used to demonstrate the GTM algorithm. The data set is artificially generated and therefore is known to lie on a lower dimensional manifold. Here we use a sub-sampled version of the data (containing 100 data points) to demonstrate the fitting of a GP-LVM with a simple radial basis function (RBF) kernel.

As we saw in Section 2.3, seeking a lower dimensional embedding with PCA is equivalent to a GP-LVM model with a linear kernel,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j + \beta^{-1} \delta_{ij},$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is the element in the i th row and the j th column of the kernel matrix \mathbf{K} and δ_{ij} is the Kronecker delta function.

For comparison we visualised the data set using several of the approaches mentioned in the introduction. In Figure 1(a) we show the first two principal components of the data. Figure 1(b) then shows the visualisation obtained using the GP-LVM with the RBF kernel,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\text{rbf}} \exp\left(-\frac{\gamma}{2} (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)\right) + \theta_{\text{bias}} + \theta_{\text{white}} \delta_{ij}.$$

To obtain this visualisation the log likelihood was optimised jointly with respect to the latent positions \mathbf{X} and the kernel parameters θ_{bias} , θ_{white} , θ_{rbf} and γ . The kernel was initialised using PCA to set \mathbf{X} , the kernel parameters were initialised as $\theta_{\text{rbf}} = \gamma = 1$ and $\theta_{\text{white}} = \theta_{\text{bias}} = \exp(-1)$.

Note that there is a redundancy in the representation between the overall scale of the matrix \mathbf{X} and the value of γ . This redundancy was removed by penalising the log likelihood (6) with half the sum of the squares of each element of \mathbf{X} : this implies we were actually seeking a MAP solution¹² with a Gaussian prior for \mathbf{X} ,

$$p(\mathbf{X}) = \prod_{n=1}^N N(\mathbf{x}_n | \mathbf{0}, \mathbf{I}).$$

The likelihood for the RBF kernel was optimised using scaled conjugate gradient (see <http://www.dcs.shef.ac.uk/~neil/gplvmapp/> for the code used).

12. Multiplying the likelihood by this prior leads to a joint distribution over data points and latent points. As a function of \mathbf{X} this joint distribution is proportional to the posterior distribution $p(\mathbf{X}|\mathbf{Y})$, therefore maximising the joint distribution is equivalent to seeking a MAP solution.

PROBABILISTIC NON-LINEAR PCA

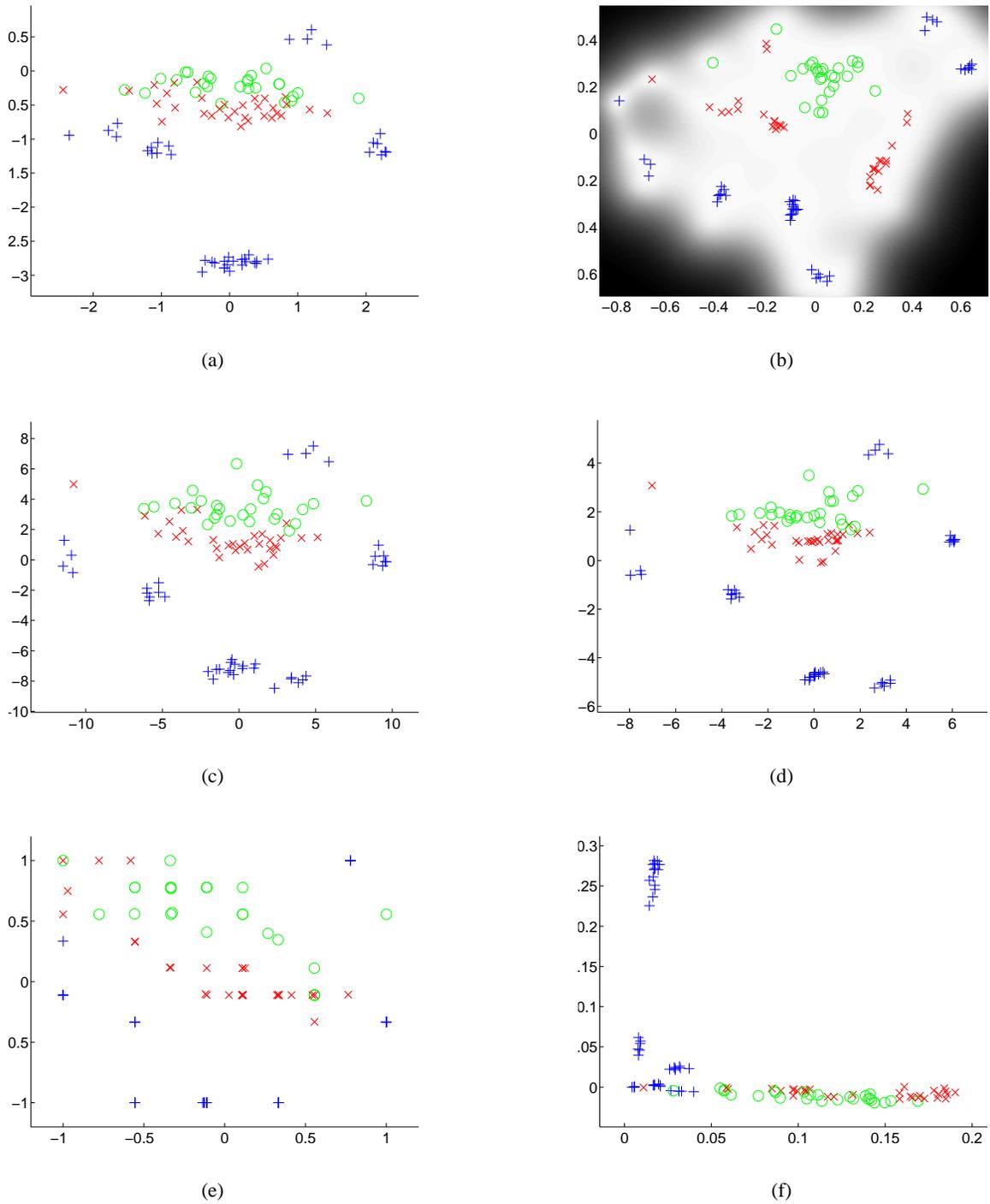


Figure 1: Visualisation of the Oil data with (a) PCA (a linear GP-LVM) and (b) A GP-LVM which uses an RBF kernel, (c) Non-metric MDS using Kruskal's stress, (d) Metric MDS using the 'Sammon Mapping', (e) GTM and (f) kernel PCA. Red crosses, green circles and blue plus signs represent stratified, annular and homogeneous flows respectively. The greyscales in plot (b) indicate the precision with which the manifold was expressed in data-space for that latent point.

Method	PCA	GP-LVM	Non-metric MDS	Metric MDS	GTM*	kernel PCA*
Errors	20	4	13	6	7	13

Table 2: Errors made by the different methods when using the latent-space for nearest neighbour classification in the latent space. Both the GTM and kernel PCA are given asterisks as the result shown is the best obtained for each method from a range of different parameterisations.

We also provide visualisations of the data using the range of algorithms we reviewed in the introduction. In Figure 1(c) we show the result of non-metric MDS using the stress criterion of Kruskal (1964). Figure 1(d) shows the result from metric MDS using the criterion of Sammon (1969). To objectively evaluate the quality of the visualisations we classified each data point according to the class of its nearest neighbour in the two dimensional latent-space supplied by each method. The errors made by such a classification are given in Table 2. For the GTM and kernel PCA some selection of parameters is required. For GTM we varied the size of the latent grid between 3×3 and 15×15 , and the number of hidden nodes in the RBF network was varied between 4 and 36. The best result was obtained for a 10×10 latent grid with 25 nodes in the RBF network, it is shown in Figure 1(e). Note the characteristic gridding effect in the GTM’s visualisation which arises from the layout of the latent points. For kernel PCA we used the RBF kernel and varied the kernel width between 0.01 and 100. The best result was obtained for a kernel width of 0.75, the associated visualisation is shown in Figure 1(f).

The gradient based optimisation of the RBF based GP-LVM’s latent-space shows results which are clearly superior (in terms of separation between the different flow phases) to those achieved by the linear PCA model. The GP-LVM approach leads to a number of errors that is the smallest of all the approaches used. Additionally the use of a Gaussian process to perform our ‘mapping’ means that we can express uncertainty about the positions of the points in the *data* space. For our formulation of the GP-LVM the level of uncertainty is shared across all D dimensions and thus may be visualised in the latent-space.

3.2.1 VISUALISING THE UNCERTAINTY

Recall that the likelihood (4) is a product of D separate Gaussian processes. In this paper we chose to retain the implicit assumption in PCA that *a priori* each dimension is identically distributed by assuming that the processes shared the same covariance/kernel function \mathbf{K} . Sharing of the covariance function also leads to an *a posteriori* shared level of uncertainty in each process. While it is possible to use different covariance functions for each dimension and may be necessary when each of the data’s attributes have different characteristics;¹³ the more constrained model implemented here allows us to visualise the uncertainty in the latent space and will be preferred for our empirical

13. A simple example of this is given by Grochow et al. (2004) with the ‘scaled GP-LVM’, where a scale parameter is associated with each dimension of the data.

studies.¹⁴ In Figure 1(b) (and subsequently) the uncertainty is visualised by varying the intensity of the background pixels. The lighter the pixel the higher the precision of the mapping.

3.2.2 COMPUTATIONAL COMPLEXITY

While the quality of the results seem good, a quick analysis of the algorithmic complexity shows that each gradient step requires an inverse of the kernel matrix (see (10)), an $O(N^3)$ operation, rendering the algorithm impractical for many data sets of interest. In the next section we will show how a practical algorithm may be developed which circumvents this problem through maximising a sparse approximation to (6).

4. A Practical Algorithm for GP-LVMs

So far we have shown that PCA can be viewed probabilistically from two perspectives, the first involves integrating latent variables and the second optimising them. Using the latter perspective we can develop a non-linear probabilistic version of PCA. Unfortunately the optimisation problem we are faced with is then non-linear and high dimensional (Nq interdependent parameters/latent-variables before we consider the parameters of the kernel). In this section we will describe an approximation that relies on a forced ‘sparsification’ of the model. The resulting computational advantages make visualisation of large numbers of data points practical. We base our approach on the informative vector machine algorithm (Lawrence et al., 2003). As we will see in Section 5, this machinery has the added advantage of allowing us to extend our non-linear PCA model to non-Gaussian noise models.

4.1 Sparsification

Kernel methods may be sped up through sparsification, *i.e.* representing the data set by a subset, I , of d points known as the *active set*. The remaining points are denoted by J . We make use of the informative vector machine (IVM) which selects points sequentially according to the reduction in the posterior process’s entropy that they induce: implementation details for the IVM algorithm are given in Lawrence et al. (2003).

A consequence of this enforced sparsification is that optimisation of the points in the active set (with $d < N$) proceeds much quicker than the optimisation of the full set of latent variables: the likelihood of the active set is given by

$$p(\mathbf{Y}_I) = \frac{1}{(2\pi)^{\frac{d}{2}} |\mathbf{K}_{I,I}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \text{tr}\left(\mathbf{K}_{I,I}^{-1} \mathbf{Y}_I \mathbf{Y}_I^T\right)\right), \quad (11)$$

which can be optimised with respect to the kernel’s parameters and \mathbf{X}_I with gradient evaluations costing $O(d^3)$ rather than the prohibitive $O(N^3)$ which would arise in the full model. The dominant cost (asymptotically) becomes that of the active selection which is $O(d^2N)$.

14. The two approaches, constraining each data direction to the same kernel and allowing each data dimension to have its own kernel are somewhat analogous to the difference between probabilistic PCA, where each output data shares a variance, and factor analysis, where each data dimension maintains its own variance.

Algorithm 1 An algorithm for visualisation with a GP-LVM.

Require: A size for the active set, d . A number of iterations, T .

Initialise \mathbf{X} through PCA.

for T iterations. **do**

 Select a new active set using the IVM algorithm.

 Optimise (11) with respect to the parameters of \mathbf{K} (and optionally the latent positions \mathbf{X}_I) using scaled conjugate gradients.

 Select a new active set.

for each point not in active set j . **do**

 Optimise (12) with respect to \mathbf{x}_j using scaled conjugate gradients.

end for

end for

4.2 Latent Variable Optimisation

We are interested in visualising all points in the data set, so while there is a significant speed advantage to selecting an active set, we still need to optimise the *inactive points*. Fortunately, active set selection allows us to optimise each of these points independently as, given a fixed active set, the individual data points are no longer interdependent. A standard result for Gaussian processes (see *e.g.* Williams, 1998) is that a point, j , from the inactive set can be shown to project into the data-space as a Gaussian distribution

$$p(\mathbf{y}_j|\mathbf{x}_j) = N(\mathbf{y}_j|\mu_j, \sigma_j^2\mathbf{I}) \quad (12)$$

whose mean is

$$\mu_j = \mathbf{Y}^T \mathbf{K}_{I,I}^{-1} \mathbf{k}_{I,j}$$

where $\mathbf{K}_{I,I}$ denotes the kernel matrix developed from the active set and $\mathbf{k}_{I,j}$ made up of rows in I from the j th column of \mathbf{K} , and the variance¹⁵ is

$$\sigma_j^2 = k(\mathbf{x}_j, \mathbf{x}_j) - \mathbf{k}_{I,j}^T \mathbf{K}_{I,I}^{-1} \mathbf{k}_{I,j}.$$

Gradients with respect to \mathbf{x}_j do not depend on other data in J , we can therefore independently optimise the likelihood of each \mathbf{y}_j with respect to corresponding \mathbf{x}_j . Thus the full set \mathbf{X}_J can be optimised with one pass through the data. The active set is then reselected, and the process is repeated again.

Algorithm 1 summarises the order in which we implemented these steps. The active set is first selected, then the kernel parameters and active set positions are optimised. The active set is then re-selected and then the latent positions of the points not in the active set are optimised. In each iteration we perform two active set selections because the choice of active set is dependent on both the kernel parameters and the latent point positions. Note also, that for some data sets (when $N \gg d$) it may not be necessary to optimise \mathbf{X}_I because the active set is regularly being reselected.

15. This fixed variance for all output dimensions is a consequence of sharing the same kernel for each output as was discussed in Section 3.2.1.

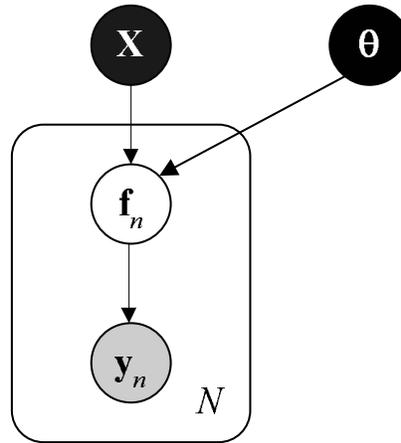


Figure 2: The Gaussian process as a latent variable model.

5. Alternative Noise Models

So far we have considered the GP-LVM for the particular case where we have Gaussian noise in each dimension with variance β^{-1} . In this section we consider extensions to this noise model. To this end we firstly reformulate our Gaussian process so that it contains an additional latent variable $\mathbf{F} = [\mathbf{f}_1 \dots \mathbf{f}_N]^T$ between \mathbf{X} and \mathbf{Y} .

$$p(\mathbf{Y}|\mathbf{X}, \theta) = \int \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{f}_n) p(\mathbf{F}|\mathbf{X}, \theta) d\mathbf{F}. \quad (13)$$

Thus far we have been considering the case where

$$p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^D N(y_{ni}|f_{ni}, \beta^{-1}),$$

it is also straightforward to realise the slightly more general case where the variance is dependent on both the data point and the output dimension,

$$p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^D N(y_{ni}|f_{ni}, \beta_{ni}^{-1}). \quad (14)$$

Our approach to different noise models will be to approximate them with a Gaussian noise model of this form (see also Csató, 2002; Minka, 2001). The noise models we consider in this paper will be independent across the dimensions,

$$p(\mathbf{y}_n|\mathbf{f}_n) = \prod_{i=1}^D p(y_{ni}|f_{ni}),$$

giving approximations of the form

$$p(y_{ni}|f_{ni}) \approx N(m_{ni}|f_{ni}, \beta_{ni}^{-1}).$$

The approximation to the noise model leads to a Gaussian approximation to the posterior distribution,

$$q(\mathbf{F}) \approx p(\mathbf{F}|\mathbf{X}, \mathbf{Y}),$$

where

$$q(\mathbf{F}) = N(\mathbf{f}|\bar{\mathbf{f}}, \Sigma)$$

where \mathbf{f} is a vector constructed by stacking the columns of \mathbf{F} , and $\bar{\mathbf{f}}$ is constructed by stacking the columns of the matrix $\bar{\mathbf{F}} = [\bar{\mathbf{f}}_1 \dots \bar{\mathbf{f}}_N]^T$. The covariance matrix has a block diagonal structure¹⁶

$$\Sigma = \begin{bmatrix} \Sigma_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Sigma_D \end{bmatrix}.$$

It can be shown (see *e.g.* Csató 2002; Minka 2001) that the parameters of the approximation are given by

$$\beta_{ni} = \frac{v_{ni}}{1 - v_{ni}\zeta_{ni}} \quad (15)$$

$$m_{ni} = \frac{g_{ni}}{v_{ni}} + \bar{f}_{ni} \quad (16)$$

where ζ_{ni} is n th diagonal element of Σ_i , $g_{ni} = \frac{\partial}{\partial \bar{f}_{ni}} \ln Z_{ni}$ and $v_{ni} = g_{ni}^2 - 2 \frac{\partial}{\partial \zeta_{ni}} \ln Z_{ni}$ where

$$Z_{ni} = \int p(y_{ni}|f_{ni}) q(\mathbf{F}) d\mathbf{F}. \quad (17)$$

To prevent cluttering our notation we have not indicated that the approximation $q(\mathbf{F})$ is typically formed in a sequential manner: its parameters $\bar{\mathbf{F}}$ and Σ change as data points are incorporated. This approach to approximating the posterior distribution is known as assumed density filtering (see Maybeck, 1979, Chapter 12 and Minka, 2001, Chapter 3).

6. Missing Values

In many applications attributes are missing for particular data points. The ability to handle these missing values in a principled way is a desirable characteristic of any algorithm. One motivation behind a probabilistic interpretation of PCA was that the resulting algorithm could handle missing data in a principled manner. This is a characteristic which the Gaussian process latent variable model shares. This should be contrasted with kernel PCA where handling missing values is not so straightforward.

Given the formalism we have described for using different noise models it is straightforward to handle a missing attribute. The corresponding variance from (14) is set to infinity by taking $\beta_{ni} = 0$.

16. For the special case of Gaussian noise with fixed variance β^{-1} (*i.e.* spherical noise) and shared kernels for each data dimension we find that these blocks are all equal. This leads to computational and memory savings. If the kernels are different or more general noise models are used the blocks will not be equal.

7. Results

In this section we present a range of empirical evaluations with different data sets, each explores a different characteristics of the GP-LVM. Note that for these visualisation algorithms over-fitting is *not* a problem as long as the latent-space is of lower dimensionality than the data-space. This is a consequence of the integration over the mapping between the latent and the data-space.

So far we have briefly considered two different kernel/covariance functions, before proceeding further we will reconsider these and introduce further kernels which will be used in the experiments that follow.

7.1 Kernels to Be Used

A Gaussian process covariance function can be developed from any positive definite kernel, new kernels can also be formed by adding kernels together. In our experiments we principally make use of three different kernel functions.

7.1.1 LINEAR KERNEL

We have already briefly discussed the linear kernel, it is simply the matrix of inner products,

$$k_{\text{lin}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\text{lin}} \mathbf{x}_i^T \mathbf{x}_j,$$

where we have introduced θ_{lin} , the process variance, which controls the scale of the output functions.

7.1.2 RBF KERNEL

We also made use of the popular RBF kernel, it leads to smooth functions that fall away to zero in regions where there is no data.

$$k_{\text{rbf}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\text{rbf}} \exp\left(-\frac{\gamma}{2} (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)\right)$$

where γ is the inverse width parameter.

7.1.3 MLP KERNEL

The MLP kernel (Williams, 1997) is derived by considering a multi-layer perceptron (MLP) with an infinite number of hidden units,

$$k_{\text{mlp}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\text{mlp}} \sin^{-1} \left(\frac{w \mathbf{x}_i^T \mathbf{x}_j + b}{\sqrt{(w \mathbf{x}_i^T \mathbf{x}_i + b + 1)(w \mathbf{x}_j^T \mathbf{x}_j + b + 1)}} \right)$$

where we call w the weight variance and b the bias variance (they have interpretations as the variances of prior distributions in the neural network model). This covariance function also leads to smooth functions, but they have an important characteristic that differentiates them from the RBF kernel: outside regions where the data lies functions will not fall to zero, but tend to remain at the same value.

7.1.4 THE NOISE TERM

In the experiments in Section 3 we also made use of a ‘white noise term’. A white noise process has a kernel of the form

$$k_{\text{white}}(\mathbf{x}_i, \mathbf{x}_j) = \theta_{\text{white}} \delta_{ij}$$

where δ_{ij} is the Kronecker delta which is zero unless $i = j$ when it takes the value 1. Note that the use of white noise in the kernel is often redundant with some parameters in the noise model, for example with a Gaussian noise model, leaving out the white noise term and setting

$$p(y_{in}|f_{in}) = N(y_{in}|f_{in}, \theta_{\text{white}})$$

is equivalent to including the white noise kernel and setting

$$p(y_{in}|f_{in}) = \lim_{\sigma^2 \rightarrow 0} N(y_{in}|f_{in}, \sigma^2).$$

In our experiments we preferred to include the noise term with the kernel as the noise level, θ_{white} , can then be jointly optimised with the kernel parameters and the latent point positions.

7.1.5 PARAMETER CONSTRAINTS AND INITIALISATION

All the kernels we have mentioned so far have parameters that need to be constrained to be positive. In our experiments this was implemented by reparameterising:

$$\theta = \ln(1 + \exp(\theta')).$$

Note that as our transformed parameter $\theta' \rightarrow -\infty$ the parameter $\theta \rightarrow 0$ and as $\theta' \rightarrow \infty$ we see that $\theta \rightarrow \theta'$.

We used a consistent initialisation of the parameters for all experiments. This was $\theta_{\text{lin}} = 1$, $\theta_{\text{rbf}} = 1$, $\gamma = 1$, $\theta_{\text{mlp}} = 1$, $w = 10$ and $b = 10$.

7.2 Overview of Experiments

For the experiments that follow we used Algorithm 1 with $T = 15$ iterations and an active set of size $d = 100$. The experiments were run on a ‘one-shot’ basis, *i.e.* each experiment was only run once with one setting of the random seed and the values of T and d given.

The remainder of this section is structured as follows, firstly, in Section 7.2.1 we revisit the oil data first introduced in Section 3.2, but with the revised algorithm which allows us to efficiently visualise all the data points. As well as comparing the sparse algorithm to the GTM and PCA we also include a full GP-LVM model. For each of the different algorithms we explore the quality of the visualisation in terms of the ease with which the different flow regimes can be separated in the embedded space. In Section 7.3.1 we turn to a much higher (256) dimension data set of hand-written digits. Again we compare the GTM and PCA with the sparse GP-LVM algorithm by seeing how well the different digits are separated in the latent-space.

In both of the preceding data sets we made use of the Gaussian noise model, our final experiment with this noise model concerns issues with initialisation. In the data sets presented above we have no simple ‘ground truth’ which the algorithm hopes to recover. In Section 7.2.3 we consider the Swiss-roll data Tenenbaum et al. (2000). For this data the ground truth is known and it turns out that using PCA to initialise the GP-LVM the ground truth is not recovered, however by initialising using

Model	PCA	Sparse GP-LVM (RBF)	GP-LVM (RBF)	Sparse GP-LVM (MLP)	GTM
Errors	162	24	1	14	11

Table 3: Number of errors for nearest neighbour classification in the latent-space for the full oil data set (1000 points).

Isomap (which is known to give the ground truth) we can recover a probabilistic representation of this data.

In Section 7.3.1 we move on to non-Gaussian data sets. We consider a binary data set of handwritten 2s. We compare a binary model with a Gaussian model and show that the binary model is more effective at reconstructing twos when pixels are obscured from the model.

7.2.1 OIL FLOW DATA

In this section we return to the twelve dimensional oil data set that we first introduced in Section 3.2. We now visualise all 1000 of the data points. For this data set we are interested in evaluating two different things: the effect of using the different non-linear kernels and the effect of the sparse GP-LVM algorithm relative to the full model.

In Figure 3(a) and (b) we present visualisations of the data using sparse GP-LVM algorithm with the RBF and MLP kernels respectively. In Figure 4(a) we show the data visualised with the non-sparse GP-LVM algorithm and in Figure 4(b) we have recreated the visualisation in (Bishop et al., 1998) which uses the GTM algorithm.

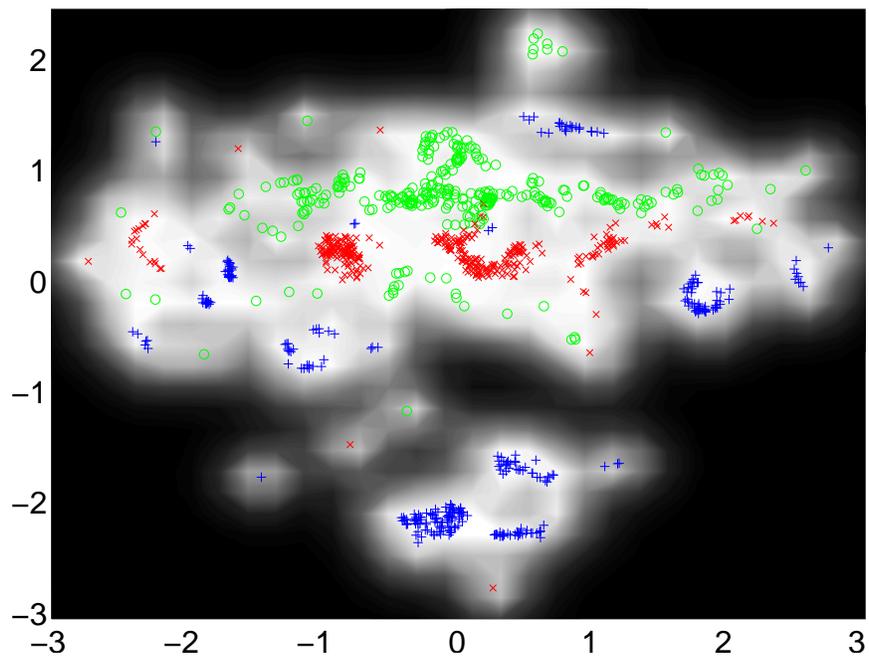
Again we considered a nearest neighbour classifier in the latent-space to quantify the quality of the visualisations.

We note that there appears to be a degradation in the quality of the GP-LVM model associated with the sparsification, in comparison to the full GP-LVM algorithm and the GTM the sparse GP-LVM performs worse.

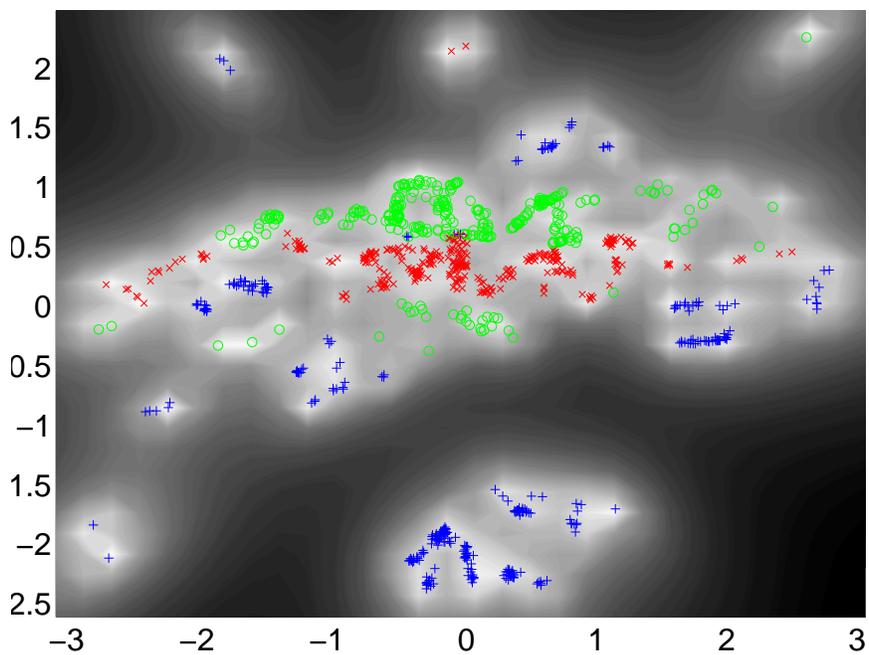
7.2.2 HANDWRITTEN DIGITS

The oil flow data has twelve attributes, twelve dimensions is too many for the structure of the data set to be visualised without resorting to displaying embedded spaces, but there are many data sets with much greater dimensionality. One popular data set for visualisation algorithms has been handwritten digits. We therefore followed Hinton and Roweis (2003) in our 2-D visualisation of a sub-set of 3000 of the digits 0-4 (600 of each digit) from a 16×16 greyscale version of the USPS digit data set (Figure 5). Again we made use of the RBF and the MLP kernel. As well as visualising with the GP-LVM we present visualisations from a GTM and PCA (Figure 6).

As for the oil data we looked for an objective assessment of the quality of the visualisation by evaluation errors on a nearest neighbour classifier in the latent-space. The performance benefits associated with the non-linear visualisations are more apparent here than they were for the oil data (Table 4). The sparse GP-LVM is once again outperformed by the GTM algorithm under this criterion. Comparison with the full GP-LVM model for this data set is not currently practical.

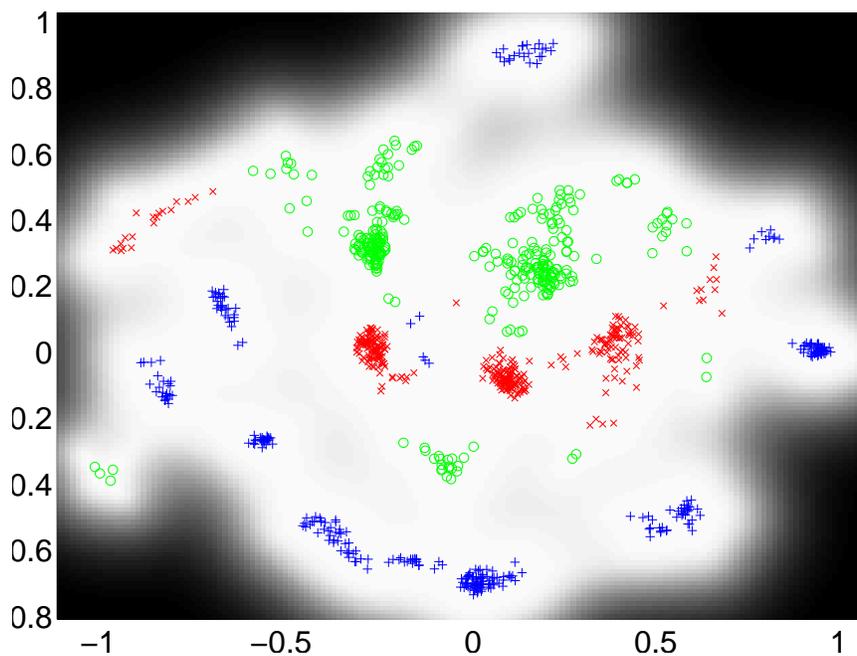


(a)

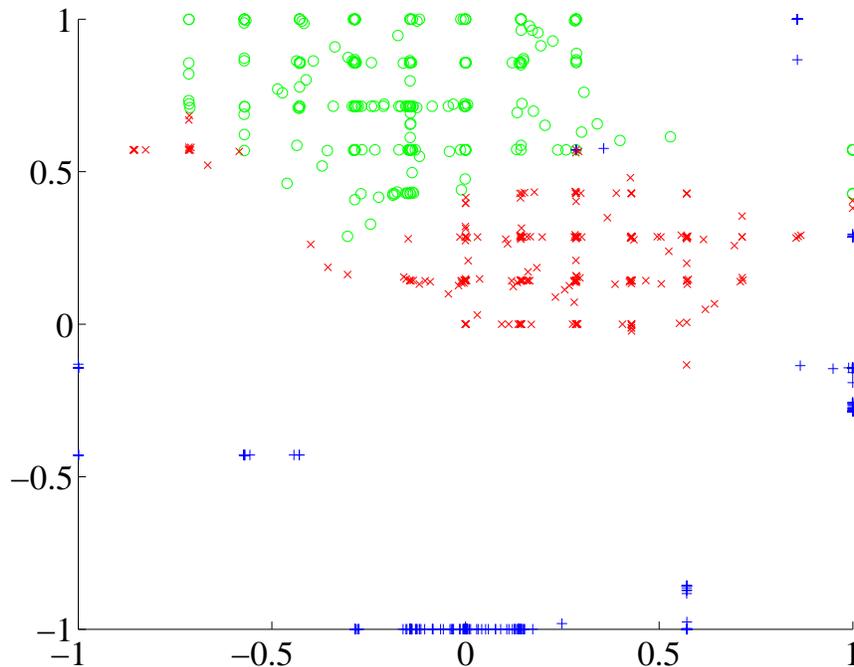


(b)

Figure 3: The full oil flow data set visualised with (a) an RBF based sparse GP-LVM, (b) an MLP based sparse GP-LVM.

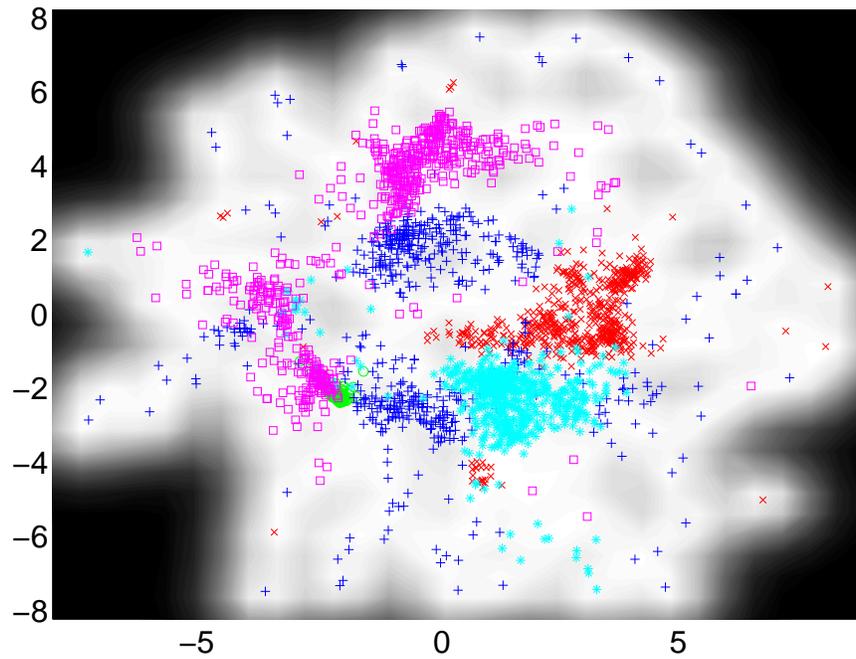


(a)

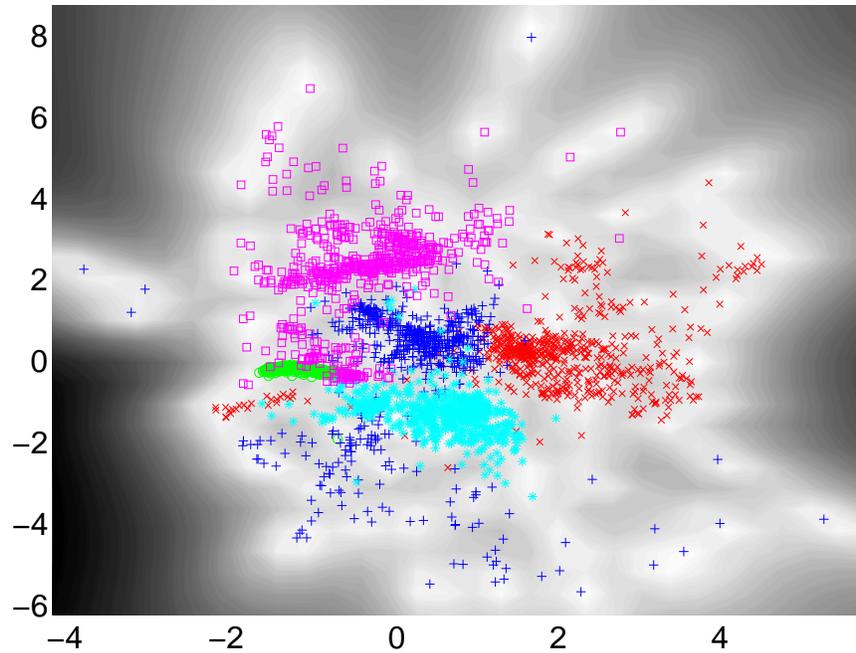


(b)

Figure 4: (a) The full GP-LVM algorithm with RBF kernel on the oil flow data. (b) GTM with 225 latent points laid out on a 15×15 grid and with 16 RBF nodes.



(a)



(b)

Figure 5: The digit images visualised in the 2-D latent-space. ‘0’ is represented by red crosses; ‘1’: green circles; ‘2’: blue pluses; ‘3’: cyan stars and ‘4’: magenta squares. (a) Visualisation using an RBF kernel. (b) Visualisation using an MLP kernel.

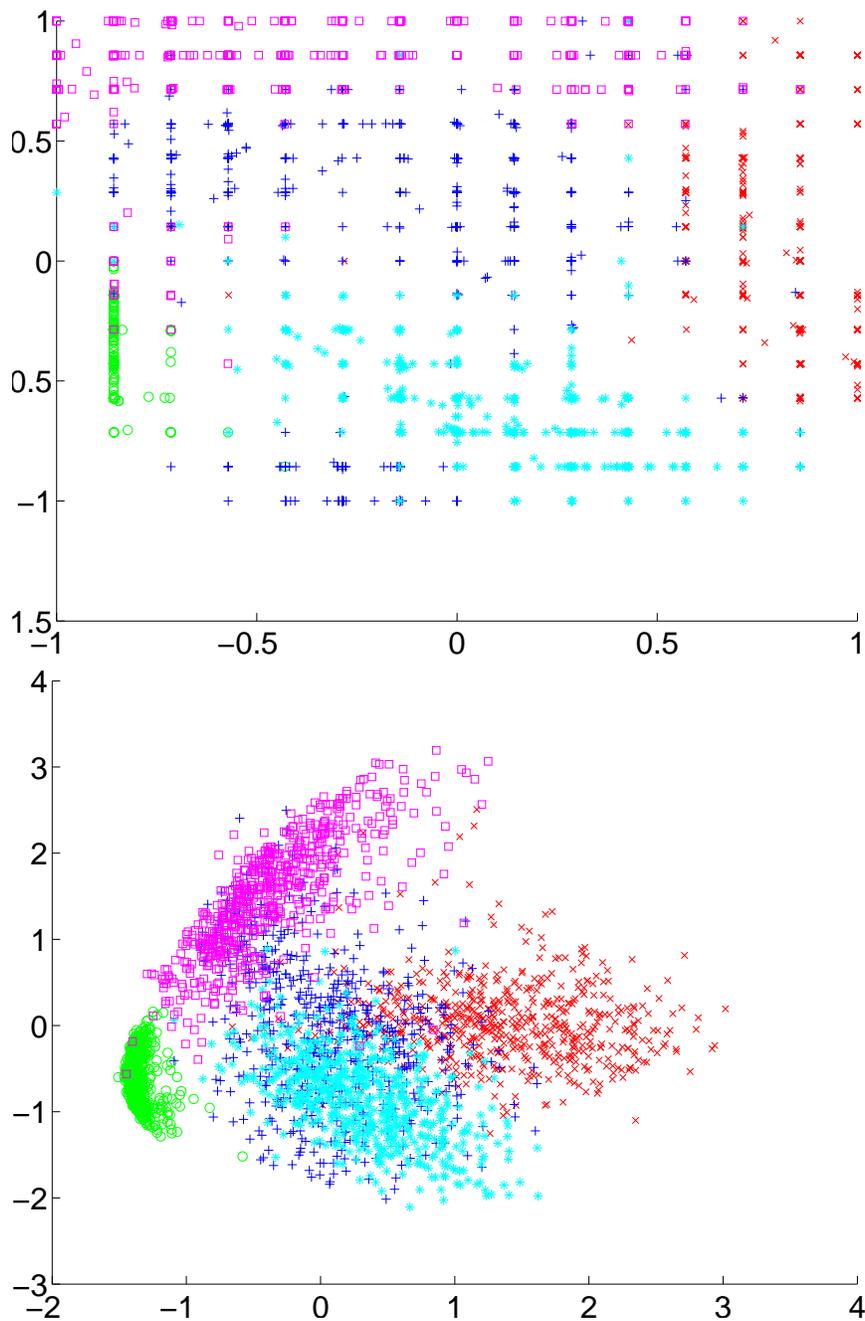


Figure 6: The digit images visualised in the 2-D latent-space. ‘0’ are red crosses, ‘1’ are green circles, ‘2’ are blue pluses, ‘3’ are cyan stars and ‘4’ are magenta squares. (a) Visualisation using the GTM algorithm. (b) Visualisation using PCA.

Model	PCA	Sparse GP-LVM (RBF)	Sparse GP-LVM (MLP)	GTM
Errors	780	208	202	158

Table 4: Errors for nearest neighbour classification in the latent-space for the digit data.

7.2.3 INITIALISATION OF THE MODEL

In the experiments we described above PCA was used to initialise the positions of the points in latent-space, however, there are data sets for which PCA can provide a poor initialisation, causing the GP-LVM to become caught in a local minima. In Figure 7(a) we show a result from modelling the ‘Swiss-roll’ data set (Tenenbaum et al., 2000, data available on line). For this data the true structure is known—the manifold is a two dimensional square twisted into a spiral along one of its dimensions and living in a three dimensional space. We follow Roweis and Saul (2000) in using colour to show the position along the sheet.

When the GP-LVM is initialised with PCA it becomes stuck in an optimum that does not recover the true embedded space. However, by initialising using the Isomap algorithm, we are able to recover the underlying structure and then provide a probabilistic description of the data through the GP-LVM (Figure 7(b)). In this way we can combine the strengths of the two different approaches—Isomap (and related proximity data based algorithms) provide a unique solution which can recover the structure of the manifold on which the data lies, the GP-LVM provides an underlying probabilistic model and an easy way to compute the mapping from the latent to the observed space. Due to the probabilistic nature of the GP-LVM we can also compare the resulting models through their log likelihood. The log likelihood of the Isomap initialised model (-45.19) is over a factor of ten smaller than that of the PCA initialised model (-534.0) further demonstrating the advantage of the Isomap initialisation for this data set.

7.3 Missing Data and Non-Gaussian Noise Models

The examples we have presented so far are for Gaussian noise models. In cases where the data is not continuous a Gaussian noise model is no longer appropriate. Non-Gaussian, *linear*, latent trait models have already been proposed (Bartholomew, 1987; Tipping, 1999), in this section we use the ADF approach described in Section 5 to explore two non-Gaussian data sets with GP-LVM models based around non-Gaussian noise models.

7.3.1 VISUALISATION OF BINARY DATA

In our first example we follow Tipping (1999) in visualising binary handwritten twos. In Figure 8 we show visualisations from an 8×8 data set derived from the USPS Cedar CD-ROM. The data contains 700 examples, these examples were taken from the complete data set of all digits used in Hinton et al. (1995). For both visualisations an RBF kernel was used in combination with a Gaussian prior over the latent-space, however the two visualisations make use of different noise models. In Figure 8(a) a Gaussian noise model was used, in Figure 8(b) a Bernoulli noise model was used.

There are certainly differences between the two visualisations in Figure 8, however we again wish to make an objective assessment of the qualities of the embedded spaces. To this end, we turned to a test set containing 400 hundred digits. For each digit in the test set we removed 20% of the pixel values. The digit was then presented to the model and its position in the embedded space

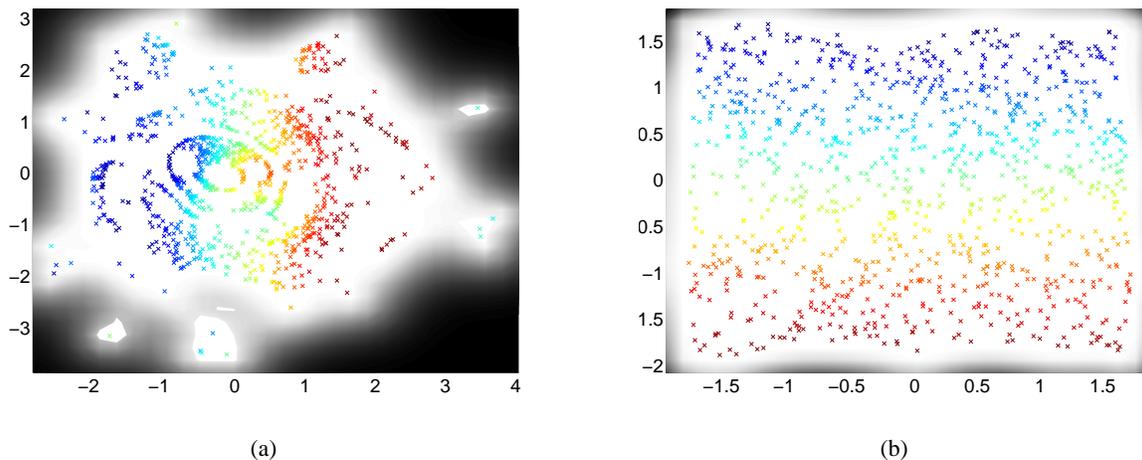


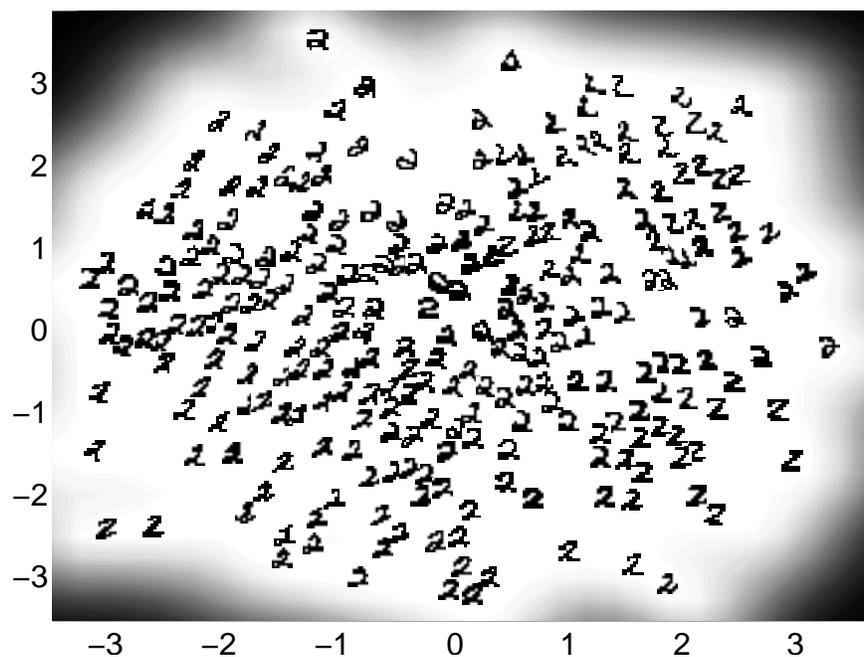
Figure 7: The effect of a poor initialisation. (a) GP-LVM initialised using PCA. The log-likelihood of the resulting model was -534.0 (b) GP-LVM initialised using Isomap. The log likelihood of the resulting model was -45.19.

Reconstruction method	pixel error rate
GP-LVM with Bernoulli noise	23.5%
GP-LVM with Gaussian noise	35.9%
Assume pixels are ‘not ink’	51.5%

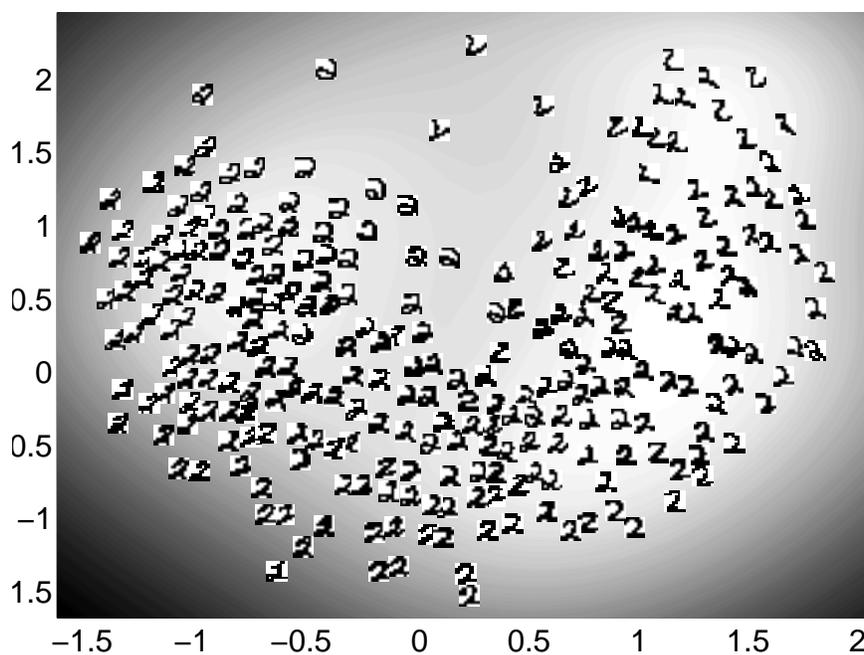
Table 5: Pixel reconstruction error rates.

optimised. The missing pixels were then filled in by using the mapping from the embedded to the data-space. Note that there can be local minima in the embedded space, we therefore optimised the embedded space location ten times with different starting positions and selected that with the largest likelihood. Since we know the original pixel values we can compute the pixel reconstruction error rate. These rates are summarised in Table 5. Results are shown for the Bernoulli noise model, the Gaussian noise model and a baseline approach (which is simply to assume that the missing pixels do not contain ink).

As might be hoped, both approaches considerably outperform the baseline approach. We also note that using the Bernoulli noise model leads to far better results than the Gaussian noise model. To illustrate the type of mistakes that are made we show some randomly sampled results in Figure 9. For each test digit we present: the original digit, an image showing which pixels are removed and reconstruction using the three methods outlined above. Note that for the GP-LVM reconstructions, particularly for the Bernoulli noise model, even when mistakes are made the resulting image often still looks like a handwritten 2.



(a)



(b)

Figure 8: The two images visualised in the 2-D latent-space. (a) Visualisation using an Gaussian noise model. (b) Visualisation using a Bernoulli noise model.

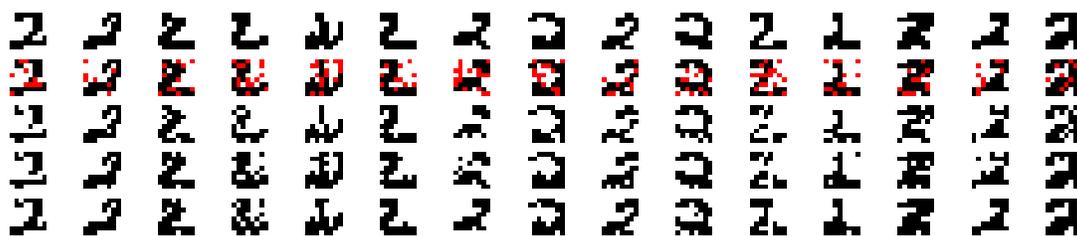


Figure 9: Randomly sampled examples from the test data for the ‘twos’ problem. *Top row*: test images from the data set of twos, *second row*: pixels removed from the test images are shown in red, *third row*: reconstruction which assumes missing pixels are ‘not ink’, *fourth row*: reconstruction by the Gaussian GP-LVM, *fifth row*: reconstruction by the binary noise model.

8. Discussion

We have presented the Gaussian process latent variable model, which is a non-linear probabilistic extension of PCA. Our experiments show that the GP-LVM is a viable alternative to other non-linear visualisation approaches for small data sets. We reviewed a practical algorithm for fitting the GP-LVM (Lawrence, 2004) in large data sets, but noted that it is associated with a degradation in performance of the method. The GP-LVM model was extended in a principled manner to take account of missing data and binary data. The advantage of explicitly modelling the data type was shown by a missing data problem in handwritten digits.

8.1 Computing the Likelihood of Test Data

One key advantage of the GP-LVM is that it is probabilistic. There is a likelihood associated with the training data. The model can be viewed as a non-parametric density estimator: the size of \mathbf{X} grows proportionally with the size of \mathbf{Y} . However this introduces particular problems when we are interested in computing the likelihood of a previously unseen (test) data point. In the traditional probabilistic PCA model when a new data point, \mathbf{y}_* , is presented its likelihood under the marginal distribution,

$$p(\mathbf{y}_* | \mathbf{W}, \beta) = N(\mathbf{y}_* | \mathbf{0}, \mathbf{W}\mathbf{W}^T + \beta^{-1}\mathbf{I}), \quad (18)$$

is easily computed. Therefore the likelihood of a previously unseen test data set is straightforward to compute. In the GP-LVM the likelihood takes a different form. The new datum has an associated latent variable, \mathbf{x}_* . The likelihood of \mathbf{y}_* , for the special case where variances over each output direction are constant, is given by

$$p(\mathbf{y}_* | \mathbf{X}, \mathbf{x}_*) = N(\mathbf{y}_* | \mu, \sigma^2), \quad (19)$$

where

$$\mu = \mathbf{Y}^T \mathbf{K}_{I,I}^{-1} \mathbf{k}_{I,*}, \quad (20)$$

$\mathbf{k}_{I,*}$ being a column vector developed from computing the elements of the kernel matrix between the active set and the new point \mathbf{x}_* . The variance is then given by

$$\sigma^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{I,*}^T \mathbf{K}_{I,I}^{-1} \mathbf{k}_{I,*}. \quad (21)$$

To determine the likelihood of the new point, we first find the MAP solution for this new latent point. The likelihood could then be approximated by computing the probability of the observed data under the distribution given by projecting the MAP solution for \mathbf{x}_* back into data-space. However, since the posterior over \mathbf{X} can be multi-modal with respect to \mathbf{x}_* , this solution will not necessarily be unique. In an ideal world, we would integrate out the latent-space to determine this marginal likelihood, and the problem with multiple modes would not arise. In practice it may be necessary to seek several modes by random restarts within the latent-space, if the likelihood is strongly peaked around each of these modes and there is a large difference between the magnitude of the two largest modes it is enough to approximate the solution with the largest mode. In other cases it may be necessary to turn to sampling methods to evaluate the likelihood.

9. Conclusions

We have presented a new class of models for probabilistic modelling and visualisation of high dimensional data. We provided theoretical groundings for these models by proving that principal component analysis is a special case. We showed there is a general objective function based on the Kullback-Leibler divergence that connects these models with proximity data based methods such as kernel PCA and multidimensional scaling. Further analysis of this objective function is expected to provide deeper insights into the behaviour of these algorithms. On real world data sets we showed that visualisations provided by the model placed related data points close to each other. We demonstrated empirically that the model performed well in traditionally difficult domains that involve missing and discrete data in high dimensions.

Our approach is related to density networks and the generative topographic mapping in that these models all provide a non-linear mapping from the embedded space to the observed space. In all these cases the embedded space is treated as a latent variable and problems of propagating distributions through the non-linear mapping are avoided by using point representations of the data within the latent space. A novel characteristic of the GP-LVM is that we can visualise the uncertainty with which the manifold is defined in the data-space.

Acknowledgments

We thank Aaron Hertzmann and his collaborators for ongoing access to their work on style based inverse kinematics, Amos Storkey for pointing out that the GP-LVM fails on the Swiss-roll data with a PCA initialisation and Michael Tipping for discussions on visualisation techniques.

Appendix A. Probabilistic Interpretations of PCA

The standard probabilistic interpretation of PCA (Tipping and Bishop, 1999) involves a likelihood,

$$p(\mathbf{Y}|\mathbf{W}, \mathbf{X}, \beta) = \prod_{n=1}^N p(\mathbf{y}_n|\mathbf{W}, \mathbf{x}_n, \beta)$$

which is taken to be Gaussian,

$$p(\mathbf{y}_n|\mathbf{W}, \mathbf{x}_n, \beta) = N(\mathbf{y}_n|\mathbf{W}\mathbf{x}_n, \beta^{-1}\mathbf{I}),$$



Figure 10: Graphical representation of (a) the standard probabilistic PCA model and (b) its dual representation which also leads to a probabilistic interpretation of PCA. The nodes are shaded to represent different treatments. *Black* shaded nodes are optimised, *white* shaded nodes are marginalised and *grey* shaded nodes are observed variables.

the prior distribution for the latent variables is then taken to be Gaussian,

$$p(\mathbf{x}_n) = N(\mathbf{x}_n | \mathbf{0}, \mathbf{I}),$$

and is duly marginalised to recover the marginal likelihood for the data,

$$p(\mathbf{Y} | \mathbf{W}, \beta) = \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{W}, \beta), \quad (22)$$

where

$$p(\mathbf{y}_n | \mathbf{W}, \beta) = N(\mathbf{y}_n | \mathbf{0}, \mathbf{W}\mathbf{W}^T + \beta^{-1}\mathbf{I}). \quad (23)$$

The structure of this model is shown graphically in Figure 10(a).

The dual representation of probabilistic PCA involves integrating out \mathbf{W} and maximising with respect to \mathbf{x}_n

$$p(\mathbf{Y} | \mathbf{X}, \beta) = \int \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}, \beta) p(\mathbf{W}) d\mathbf{W}.$$

By first specifying a prior distribution,

$$p(\mathbf{W}) = \prod_i N(\mathbf{w}_i | \mathbf{0}, \mathbf{I})$$

where \mathbf{w}_i is the i th row of the matrix \mathbf{W} , and then integrating over \mathbf{W} we obtain a marginalised likelihood for \mathbf{Y} ,

$$p(\mathbf{Y} | \mathbf{X}, \beta) = \frac{1}{(2\pi)^{\frac{DN}{2}} |\mathbf{K}|^{\frac{D}{2}}} \exp\left(-\frac{1}{2} \text{tr}(\mathbf{K}^{-1} \mathbf{Y} \mathbf{Y}^T)\right), \quad (24)$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}$ and $\mathbf{X} = [\mathbf{x}_1^T \dots \mathbf{x}_N^T]^T$. The structure of this model is shown in 10(b). Note that by taking $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \beta^{-1}\mathbf{I}$ we and substituting (23) into (22) as

$$p(\mathbf{Y} | \mathbf{X}, \beta) = \frac{1}{(2\pi)^{\frac{DN}{2}} |\mathbf{C}|^{\frac{N}{2}}} \exp\left(-\frac{1}{2} \text{tr}(\mathbf{C}^{-1} \mathbf{Y}^T \mathbf{Y})\right),$$

which highlights to a greater extent the duality between (24) and (22). Optimisation of (24) is clearly highly related to optimisation of (22). Tipping and Bishop (1999) showed how to optimise (22), in the next section we review this optimisation for DPPCA, but generalise it slightly so that it applies for any symmetric matrix \mathbf{S} , rather than only the inner product matrix $\mathbf{Y}\mathbf{Y}^T$. Thereby the derivation also covers the kernel PCA and multidimensional scaling cases outlined in Section 2.6.

Appendix B. Optimisation of Dual PCA, KPCA and MDS Objective functions

Maximising (24) is equivalent to minimising

$$L = \frac{N}{2} \ln 2\pi + \frac{1}{2} \ln |\mathbf{K}| + \frac{1}{2} \text{tr}(\mathbf{K}^{-1}\mathbf{S}), \quad (25)$$

where $\mathbf{S} = D^{-1}\mathbf{Y}\mathbf{Y}^T$. The derivation that follows holds regardless of the form of \mathbf{S} and therefore also applies to the objective function outlined in Section 2.6. However, \mathbf{S} needn't be constrained to this form, we outlined an objective function (for kernel PCA) in where \mathbf{S} was any positive definite kernel.

The gradient of the likelihood with respect to \mathbf{X} can be found as

$$\frac{\partial L}{\partial \mathbf{X}} = -\mathbf{K}^{-1}\mathbf{S}\mathbf{K}^{-1}\mathbf{X} + \mathbf{K}^{-1}\mathbf{X},$$

setting the equation to zero and pre-multiplying by \mathbf{K} gives

$$\mathbf{S} [\beta^{-1}\mathbf{I} + \mathbf{X}\mathbf{X}^T]^{-1} \mathbf{X} = \mathbf{X}.$$

We substitute \mathbf{X} with its singular value decomposition, $\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T$, giving

$$\mathbf{S}\mathbf{U} [\mathbf{L} + \beta^{-1}\mathbf{L}^{-1}]^{-1} \mathbf{V}^T = \mathbf{U}\mathbf{L}\mathbf{V}^T$$

Right multiplying both sides by \mathbf{V} (note that the solution is invariant to \mathbf{V}) we have, after some rearrangement,

$$\mathbf{S}\mathbf{U} = \mathbf{U}(\beta^{-1}\mathbf{I} + \mathbf{L}^2),$$

which, since $(\beta^{-1}\mathbf{I} + \mathbf{L}^2)$ is diagonal can be solved by an eigenvalue problem where \mathbf{U} are eigenvectors of \mathbf{S} and $\Lambda = (\beta^{-1}\mathbf{I} + \mathbf{L}^2)$ are the eigenvalues. This implies that the elements from the diagonal of \mathbf{L} are given by

$$l_i = (\lambda_i - \beta^{-1})^{\frac{1}{2}}. \quad (26)$$

B.1 The Retained Eigenvalues

The natural follow up question is which of the N possible eigenvalues/vector pairs should be retained? For convenience let us ignore our previously defined ordering of the eigenvalues in terms of their magnitude and assume that we keep the first q eigenvalues.

First note that

$$\mathbf{K} = \mathbf{U} [\mathbf{L}^2 + \beta^{-1}\mathbf{I}] \mathbf{U}^T$$

where \mathbf{U} is all the eigenvectors of \mathbf{S} . The full KL divergence is

$$\begin{aligned} \text{KL}(\mathbf{S}||\mathbf{K}) &= \frac{1}{2} \ln |\mathbf{K}| - \frac{1}{2} \ln |\mathbf{S}| + \frac{1}{2} \text{tr}(\mathbf{K}^{-1}\mathbf{S}) - \frac{N}{2} \\ &= \frac{1}{2} \sum_{i=1}^q \ln \lambda_i - \frac{N-q}{2} \ln \beta - \frac{1}{2} \sum_{i=1}^N \ln \lambda_i + \frac{1}{2} \text{tr}([\mathbf{L}^2 + \beta^{-1}\mathbf{I}]^{-1} \Lambda) \\ &= -\frac{1}{2} \sum_{i=q+1}^N \ln \lambda_i - \frac{N-q}{2} \ln \beta - \frac{N-q}{2} + \frac{\beta}{2} \sum_{i=q+1}^N \lambda_i \end{aligned}$$

where we have used the fact that $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^T$. Differentiating with respect to β and setting the result to zero to obtain a fixed point equation then gives

$$\beta = \frac{N-q}{\sum_{i=q+1}^N \lambda_i}$$

which when substituted back leads to

$$\text{KL}(\mathbf{S}||\mathbf{K}) = \frac{N-q}{2} \left(\ln \frac{\sum_{i=q+1}^N \lambda_i}{N-q} - \frac{1}{N-q} \sum_{i=q+1}^N \ln \lambda_i \right), \quad (27)$$

which is recognised as the difference between the log ratio of the arithmetic and geometric means of the discarded eigenvalues. This difference will be zero if and only if the discarded eigenvalues are constant (when the arithmetic and geometric means become equal) otherwise it is positive. The difference is minimised by ensuring that the eigenvalues we discard are adjacent to each other in terms of magnitude.

Which eigenvalues should we then discard? From (26) we note that the retained eigenvalues must be larger than β , otherwise l_i will be complex. The only way this can be true is if we discard the smallest $N-q$ eigenvalues, as retaining any others would force at least one eigenvalue of \mathbf{X} to be negative.

Appendix C. Equivalence of Eigenvalue Problems

In this section we review the equivalence of the eigenvalue problems associated with DPPCA and PPCA. For DPPCA the eigenvalue problem is of the form

$$\mathbf{Y}\mathbf{Y}^T\mathbf{U} = \mathbf{U}\Lambda.$$

Premultiplying by \mathbf{Y}^T then gives

$$\mathbf{Y}^T\mathbf{Y}\mathbf{Y}^T\mathbf{U} = \mathbf{Y}^T\mathbf{U}\Lambda \quad (28)$$

Since the \mathbf{U} are the eigenvectors of $\mathbf{Y}\mathbf{Y}^T$ (see the previous section) the matrix $\mathbf{U}^T\mathbf{Y}\mathbf{Y}^T\mathbf{U} = \Lambda$, therefore matrix $\mathbf{U}' = \mathbf{Y}^T\mathbf{U}\Lambda^{-\frac{1}{2}}$ is orthonormal. Post multiplying both sides of (28) by $\Lambda^{-\frac{1}{2}}$ gives

$$\mathbf{Y}^T\mathbf{Y}\mathbf{U}' = \mathbf{U}'\Lambda$$

which is recognised as the form of the eigenvalue problem associated with PPCA, where the eigenvectors of $\mathbf{Y}^T\mathbf{Y}$ are given by $\mathbf{U}' = \mathbf{Y}^T\mathbf{U}\Lambda^{-\frac{1}{2}}$ and the eigenvalues are given by Λ (as they were for DPPCA).

References

- David J. Bartholomew. *Latent Variable Models and Factor Analysis*. Charles Griffin & Co. Ltd, London, 1987.
- Alexander Basilevsky. *Statistical Factor Analysis and Related Methods*. Wiley, New York, 1994.
- Christopher M. Bishop. Bayesian PCA. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 482–388, Cambridge, MA, 1999. MIT Press.
- Christopher M. Bishop and Gwilym D. James. Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research*, A327:580–593, 1993.
- Christopher M. Bishop, Marcus Svensén, and Christopher K. I. Williams. A fast EM algorithm for latent variable density models. In D. S. Touretzky, Michael C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 465–471. MIT Press, 1996.
- Christopher M. Bishop, Marcus Svensén, and Christopher K. I. Williams. GTM: a principled alternative to the Self-Organizing Map. In *Advances in Neural Information Processing Systems*, volume 9, pages 354–360. MIT Press, 1997.
- Christopher M. Bishop, Marcus Svensén, and Christopher K. I. Williams. GTM: the Generative Topographic Mapping. *Neural Computation*, 10(1):215–234, 1998.
- Lehel Csató. *Gaussian Processes — Iterative Sparse Approximations*. PhD thesis, Aston University, 2002.
- Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popovic. Style-based inverse kinematics. In *ACM Transactions on Graphics (SIGGRAPH 2004)*, 2004.
- Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161, 1995.
- Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In Sue Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 857–864, Cambridge, MA, 2003. MIT Press.
- Antti Honkela and Harri Valpola. Unsupervised variational Bayesian learning of nonlinear models. In Lawrence Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 593–600, Cambridge, MA, 2005. MIT Press.
- Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- Joseph B. Kruskal. Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–28, 1964.
- Solomon Kullback and Richard A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

- Neil D. Lawrence. Gaussian process models for visualisation of high dimensional data. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 329–336, Cambridge, MA, 2004. MIT Press.
- Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Sue Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 625–632, Cambridge, MA, 2003. MIT Press.
- David Lowe and Michael E. Tipping. Feed-forward neural networks and topographic mappings for exploratory data analysis. *Neural Computing and Applications*, 4(83), 1996.
- David B. MacKay and J. L. Zinnes. A probabilistic model for the multidimensional scaling of proximity and preference data. *Marketing Sciences*, 5:325–334, 1986.
- David J. C. MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, A*, 354(1):73–80, 1995.
- Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley and Sons, Chichester, West Sussex, 2nd edition, 1999.
- Kantilal V. Mardia, John T. Kent, and John M. Bibby. *Multivariate analysis*. Academic Press, London, 1979.
- Peter S. Maybeck. *Stochastic Models, Estimation and Control, Volume 1*, volume 141 of *Mathematics in Science and Engineering*. Academic Press, New York, NY, 1979. ISBN 0-12-4807011.
- Thomas P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.
- Martin F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.
- Man-Suk Oh and Adrian E. Raftery. Bayesian multidimensional scaling and choice of dimension. *Journal of the American Statistical Association*, 96:1031–1044, 2001.
- Anthony O’Hagan. Some Bayesian numerical analysis. In José M. Bernardo, James O. Berger, A. Phillip Dawid, and Adrian F. M. Smith, editors, *Bayesian Statistics 4*, pages 345–363, Valencia, 1992. Oxford University Press.
- Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
- John W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.
- Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
- Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2001.

- Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
- Joshua B. Tenenbaum, Virginia de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- Michael E. Tipping. *Topographic Mappings and Feed-Forward Neural Networks*. PhD thesis, Aston University, Aston Street, Birmingham B4 7ET, U.K., 1996.
- Michael E. Tipping. Probabilistic visualisation of high-dimensional binary data. In Michael J. Kearns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 592–598, Cambridge, MA, 1999. MIT Press.
- Michael E. Tipping. Sparse kernel principal component analysis. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 633–639, Cambridge, MA, 2001. MIT Press.
- Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, B*, 6(3):611–622, 1999.
- Warren S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.
- Christopher K. I. Williams. Computing with infinite networks. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, Cambridge, MA, 1997. MIT Press.
- Christopher K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Michael I. Jordan, editor, *Learning in Graphical Models*, volume 89 of *Series D: Behavioural and Social Sciences*, Dordrecht, The Netherlands, 1998. Kluwer.
- Christopher K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 675–681, Cambridge, MA, 2001. MIT Press.

A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data

Rie Kubota Ando

*IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.*

RIE1@US.IBM.COM

Tong Zhang

*Yahoo Research
New York, NY, U.S.A.*

TZHANG@YAHOO-INC.COM

Editor: Peter Bartlett

Abstract

One of the most important issues in machine learning is whether one can improve the performance of a supervised learning algorithm by including unlabeled data. Methods that use both labeled and unlabeled data are generally referred to as semi-supervised learning. Although a number of such methods are proposed, at the current stage, we still don't have a complete understanding of their effectiveness. This paper investigates a closely related problem, which leads to a novel approach to semi-supervised learning. Specifically we consider learning predictive structures on hypothesis spaces (that is, what kind of classifiers have good predictive power) from multiple learning tasks. We present a general framework in which the structural learning problem can be formulated and analyzed theoretically, and relate it to learning with unlabeled data. Under this framework, algorithms for structural learning will be proposed, and computational issues will be investigated. Experiments will be given to demonstrate the effectiveness of the proposed algorithms in the semi-supervised learning setting.

1. Introduction

In machine learning applications, one can often find a large amount of unlabeled data without difficulty, while labeled data are costly to obtain. Therefore a natural question is whether we can use unlabeled data to build a more accurate classifier, given the same amount of labeled data. This problem is often referred to as semi-supervised learning.

In general, semi-supervised learning algorithms use both labeled and unlabeled data to train a classifier. Although a number of methods have been proposed, their effectiveness is not always clear. For example, Vapnik introduced the notion of transductive inference (Vapnik, 1998), which may be regarded as an approach to semi-supervised learning. Although some success has been reported (e.g., see Joachims, 1999), there has also been criticism pointing out that this method may not behave well under some circumstances (Zhang and Oles, 2000). Another popular semi-supervised learning method is co-training (Blum and Mitchell, 1998), which is related to the bootstrap method used in some NLP applications (Yarowsky, 1995) and to EM (Nigam et al., 2000). The basic idea is to label part of unlabeled data using a high precision classifier, and then put the “automatically-

labeled” data into the training data. However, it was pointed out by Pierce and Cardie (2001) that this method may degrade the classification performance when the assumptions of the method are not satisfied (that is when noise is introduced into the labels through non-perfect classification). This phenomenon is also observed in some of our experiments reported in Section 5.

Another approach to semi-supervised learning is based on a different philosophy. The basic idea is to define good functional structures using unlabeled data. Since it does not bootstrap labels, there is no label noise which can potentially corrupt the learning procedure. An example of this approach is to use unlabeled data to create a data-manifold (graph structure), on which proper smooth function classes can be defined (Szummer and Jaakkola, 2002; Zhou et al., 2004; Zhu et al., 2003). If such smooth functions can characterize the underlying classifier very well, then one is able to improve the classification performance.

It is worth pointing out that smooth function classes based on graph structures do not necessarily have good predictive power. Therefore a more general approach, based on the same underlying principle, is to directly learn a good underlying smooth function class (that is, what good classifiers are like). If the learning procedure takes advantage of unlabeled data, then we obtain a semi-supervised learning method that is specifically aimed at finding structures with good predictive power.

This motivates the general framework we are going to develop in this paper. That is, we want to learn some underlying predictive functional structures (smooth function classes) that can characterize what good predictors are like. We call this problem *structural learning*. Our key idea is to learn such structures by considering multiple prediction problems simultaneously. At the intuitive level, when we observe multiple predictors for different problems, we have a good sample of the underlying predictor space, which can be analyzed to find the common structures shared by these predictors. Once important predictive structures on the predictor space are discovered, we can then use the information to improve upon each individual prediction problem. A main focus of this paper is to formalize this intuitive idea and analyze properties of structural learning more rigorously.

The idea that one can benefit by considering multiple problems together has appeared in the statistical literature. In particular, Bayesian hierarchical modeling is motivated from the same principle. However, the framework developed in this paper is under the frequentist setting, and the most relevant statistical studies are shrinkage methods in multiple-output linear models (see Section 3.4.6 of Hastie et al., 2001). In particular, the algorithm proposed in Section 3 has a form similar to a shrinkage method proposed by Breiman and Friedman (1997). However, the framework presented here (as well as the specific algorithm in Section 3) is more general than the earlier statistical studies. In the machine learning literature, related work is sometime referred to as *multi-task learning*, for example, see (Baxter, 2000; Ben-David and Schuller, 2003; Caruana, 1997; Evgeniou and Pontil, 2004; Micchelli and Pontil, 2005) and references therein. We shall call our procedure structural learning since it is a more accurate description of what our method does in the semi-supervised learning setting. That is, we transfer the predictive structure learned from multiple tasks (on unlabeled data) to the target supervised problem. In the literature, this idea is also referred to as *inductive transfer*. The success of this approach depends on whether the learned structure is helpful for the target supervised problem.

It follows that although this work is motivated by semi-supervised learning, the general structural learning (or multi-task learning) problem considered in the paper is of independent interest. For semi-supervised learning, as we shall show later, the multiple prediction problems needed for structural learning can be generated from unlabeled data. However, the basic framework can also be applied to other applications where we have multiple prediction problems that are not necessarily derived from unlabeled data (as in the earlier statistical and machine learning studies). Because of this, the first part of the paper focuses on the development of a general structural learning paradigm as well as our algorithm. The main implication is that one can reliably learn a good underlying structure if it is shared by multiple prediction problems. In the second part, we shall demonstrate how to apply the idea of learning structure to semi-supervised learning, and demonstrate the effectiveness of the proposed method in this context.

A short version of this paper, mainly reporting some empirical results, appeared in ACL (Ando and Zhang, 2005). This version includes a more complete derivation of the proposed algorithm, with theoretical analysis and several additional experimental results. In Section 2, we formally introduce the structural learning problem under the framework of standard machine learning. We then propose a specific algorithm that finds a common low-dimensional feature space shared by the multi-problems. The algorithm will be studied in Section 3, with theoretical analysis given in Appendix A. Section 4 shows how to apply structural learning in the context of semi-supervised learning. The basic idea is to use unlabeled data to generate auxiliary prediction problems that are useful for discovering important predictive structures. Such structures can then be estimated using the algorithm developed in Section 3. We will also give intuitive justifications on why the structure shared by the artificially created auxiliary problems is helpful for the supervised problem. Experiments are provided in Section 5 to illustrate the effectiveness of the algorithm proposed in Section 3 on several semi-supervised tasks. Section 6 presents a high level summary of the main ideas developed in the paper.

2. The Structural Learning Problem

This section introduces the problem of learning predictive functional structures. Although related ideas have been explored in some earlier statistical and machine learning studies, for completeness, we shall include a self-contained description. The framework considered here will be the basis of our algorithm presented in Section 3.

2.1 Supervised Learning

In the standard formulation of supervised learning, we seek a predictor that maps an input vector $\mathbf{x} \in \mathcal{X}$ to the corresponding output $y \in \mathcal{Y}$. Usually, one selects the predictor from a set \mathcal{H} of functions based on a finite set of training examples $\{(\mathbf{X}_i, Y_i)\}$ that are independently generated according to some unknown probability distribution \mathcal{D} . The set \mathcal{H} , often called the *hypothesis space*, consists of functions from \mathcal{X} to \mathcal{Y} that can be used to predict the output in \mathcal{Y} of an input datum in \mathcal{X} . Our goal is to find a predictor f so that its error with respect to \mathcal{D} is as small as possible. In this paper, we assume that the quality of the

predictor p is measured by the expected loss with respect to \mathcal{D} :

$$R(f) = \mathbf{E}_{\mathbf{X}, Y} L(f(\mathbf{X}), Y).$$

Given a set of training data, a frequently used method for finding a predictor $\hat{f} \in \mathcal{H}$ is to minimize the empirical error on the training data (often called *empirical risk minimization* or ERM):

$$\hat{f} = \arg \min_{f \in \mathcal{H}} \sum_{i=1}^n L(f(\mathbf{X}_i), Y_i).$$

It is well-known that with a fixed sample size, the smaller the hypothesis space \mathcal{H} , the easier it is to learn the best predictor in \mathcal{H} . The error caused by learning the best predictor from finite sample is called the *estimation error*. However, the smaller the hypothesis space \mathcal{H} , the less accurate the best predictor in \mathcal{H} becomes. The error caused by using a restricted \mathcal{H} is often referred to as the *approximation error*. In supervised learning, one needs to select the size of \mathcal{H} to balance the trade-off between approximation error and estimation error. This is typically done through model selection, where we learn a set of predictors from a set of candidate hypothesis spaces \mathcal{H}_θ , and then pick the best choice on a validation set.

2.2 Learning Good Hypothesis Spaces

In practice, a good hypothesis space should have a small approximation error and a small estimation error. The problem of choosing a good hypothesis space is central to the performance of the learning algorithm, but often requires specific domain knowledge or assumptions of the world.

Assume that we have a set of candidate hypothesis spaces. If one only observes a single prediction problem $\mathcal{X} \rightarrow \mathcal{Y}$ on the underlying domain \mathcal{X} , then a standard approach to hypothesis space selection (or model selection) is by cross validation. If one observes multiple prediction problems on the same underlying domain, then it is possible to make better estimate of the underlying hypothesis space by considering these problems simultaneously.

We now describe a simple model for structural learning, which is the foundation of this paper. A similar point of view can also be found in (Baxter, 2000). Consider m learning problems indexed by $\ell \in \{1, \dots, m\}$, each with n_ℓ samples $(\mathbf{X}_i^\ell, Y_i^\ell)$ indexed by $i \in \{1, \dots, n_\ell\}$, which are independently drawn from a distribution \mathcal{D}_ℓ . For each problem ℓ , assume that we have a set of candidate hypothesis spaces $\mathcal{H}_{\ell, \theta}$ indexed by a common structural parameter $\theta \in \Gamma$ that is shared among the problems.

Now, for the ℓ -th problem, we are interested in finding a predictor $f_\ell : \mathcal{X} \rightarrow \mathcal{Y}$ in $\mathcal{H}_{\ell, \theta}$ that minimizes the expected loss over \mathcal{D}_ℓ . For notational simplicity, we assume that the problems have the same loss function (although the requirement is not essential in our analysis). Given a fixed structural parameter θ , the predictor for each problem can be estimated using empirical risk minimization (ERM) over the hypothesis space $\mathcal{H}_{\ell, \theta}$:

$$\hat{f}_{\ell, \theta} = \arg \min_{f \in \mathcal{H}_{\ell, \theta}} \sum_{i=1}^{n_\ell} L(f(\mathbf{X}_i^\ell), Y_i^\ell), \quad (\ell = 1, \dots, m). \quad (1)$$

The purpose of structural learning is to find an optimal structural parameter θ such that the expected risks of the predictors $\hat{f}_{\ell, \theta}$ (each with respect to the corresponding distribution \mathcal{D}_ℓ), when averaged over $\ell = 1, \dots, m$, are minimized.

If we use cross-validation for structural parameter selection, then we can immediately notice that a more stable estimate of the optimal θ can be obtained by considering multiple learning tasks together. In particular, if for each problem ℓ , we have a validation set $(\bar{\mathbf{X}}_j^\ell, \bar{Y}_j^\ell)$ for $j = 1, \dots, \bar{n}_\ell$, then for structural learning, the total number of validation data is $\sum_{\ell=1}^m \bar{n}_\ell$. Therefore effectively, we have more data for the purpose of selecting the optimal shared hypothesis space structure. This implies that even if the sample sizes are small for the individual problems, as long as m is large, we are able to find the optimal θ accurately. A PAC style analysis will be provided in Appendix A, where we can state a similar result without cross-validation.

In general, we expect that the hypothesis space $\mathcal{H}_{\ell, \theta}$ determines the functional structure of the learned predictor. The θ parameter can be a continuous parameter that encodes our assumption of what a good predictor should be like. If we have a large parameter space, then we can explore many possible functional structures. This argument (more rigorous results are given in Appendix A) implies that it is possible to discover the optimal shared structure when the number of problems m is large.

2.3 Good Structures on the Input Space

The purpose of this section is to provide an intuitive discussion on why in principle, there exist good functional structures (good hypothesis spaces) shared by multiple tasks. Conceptually, we may consider the simple case $\mathcal{H}_{\ell, \theta} = \mathcal{H}_\theta$, where different problems share exactly the same underlying hypothesis space.

Given an arbitrary input space \mathcal{X} without any known structure, we argue that it is often possible to learn what a good predictor looks like from multiple prediction problems. The key reason is that in practice, not all predictors are equally good (or equally likely to be observed). In real world applications, one usually observes “smooth” predictors where the smoothness is with respect to a certain intrinsic underlying distance on the input space. In general, if two points are close in this intrinsic distance, then the values that a good predictor produces at these points are also likely to be similar. In particular, completely random predictors are likely to be bad predictors, and are rarely observed in practical applications.

In machine learning, the smoothness condition is often enforced by the hypothesis space we select. For example, kernel methods constrain the smoothness of a function using a certain reproducing kernel Hilbert space (RKHS) norm. For such functions (in a RKHS), closeness of two points under a certain metric often implies closeness in predictive values. One may also consider more complicated smoothness conditions that explore the observed data-manifold (e.g. graph-based semi-supervised learning methods mentioned in the introduction). Such a smoothness condition will be useful if it correlates well with predictive ability.

In general, a good distance measure on \mathcal{X} induces a good hypothesis space which enforces smoothness with respect to the underlying distance. However, in reality, it is often not clear what is the best distance measure in the underlying space. For example, in natural language processing, the space \mathcal{X} consists of discrete points such as words, for which no appropriate distance can be easily defined. Even for continuous vector-valued input points, it is difficult to justify that the Euclidean distance is better than something else. Even after

a good distance function can be selected, it is not clear whether we can define appropriate smoothness conditions with respect to the distance.

If we observe multiple tasks, then important common structures can be discovered simply by analyzing the multiple predictors learned from the data. If these tasks are very similar to the actual learning task which we are interested in, then we can benefit significantly from the discovered structures. Even if the tasks are not directly related, the discovered structures can still be useful. This is because in general, predictors tend to share similar smoothness conditions with respect to a certain distance that is intrinsic to the underlying input space.

As an example to illustrate the main argument graphically, we consider a discrete input space of six points $\mathcal{X} = \{A, B, C, D, E, F\}$. Assume we obtain estimates of three functions from three different prediction problems, and plot the obtained function values against the input points in Figure 1. In this example, we can notice that function values at points $A, C,$ and D are similar, while function values at points F and E are similar. Therefore by observing the estimated functions, we may conclude that under some intrinsic distance metric on \mathcal{X} , points $A, C,$ and D are “close” to each other, and points E and F are “close” to each other. A good function on \mathcal{X} should be smooth with respect to this intrinsic distance. We will come back to the argument presented in this section using text data as a more concrete example, when we discuss semi-supervised learning in Section 4.

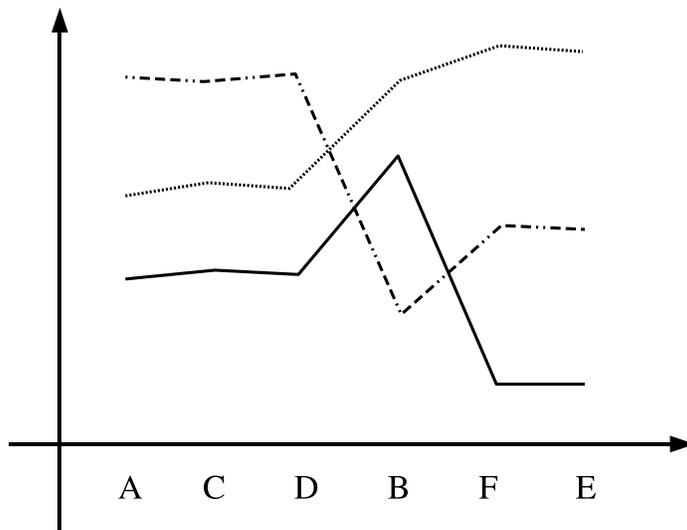


Figure 1: An Illustration of Discovering Functional Structure From Multiple Prediction Tasks

2.4 A More Abstract Form of Structural Learning

We may also pose structural learning in a slightly more abstract form, which is useful when we don't use empirical risk minimization as the learner.

Assume that for each problem ℓ , we are given a learning algorithm \mathcal{A}_ℓ that takes a set of training samples $S_\ell = \{(\mathbf{X}_i^\ell, Y_i^\ell)\}_{i=1, \dots, n_\ell}$ and a structural parameter $\theta \in \Gamma$, and produce a predictor $\hat{f}_{\ell, \theta}$: $\hat{f}_{\ell, \theta} = \mathcal{A}_\ell(S_\ell, \theta)$. Note that if the algorithm estimates the predictor from a hypothesis space $\mathcal{H}_{\ell, \theta}$ by empirical risk minimization, then we have $\hat{f}_{\ell, \theta} \in \mathcal{H}_{\ell, \theta}$.

Assume further that there is a procedure that estimates the performance of the learned predictor $\hat{f}_{\ell, \theta}$ using possibly additional information T_ℓ (which for example, could be a validation set) as $\mathcal{O}_\ell(S_\ell, T_\ell, \theta)$. Then in structural learning, we find $\hat{\theta}$ by using a regularized estimator

$$\hat{\theta} = \arg \min_{\theta \in \Gamma} \left[r(\theta) + \sum_{\ell=1}^m \mathcal{O}_\ell(S_\ell, T_\ell, \theta) \right], \quad (2)$$

where $r(\theta)$ is a regularization parameter that encodes our belief on what θ value is preferred. The number of problems m behaves like the sample size in standard learning. This is our fundamental estimation method for structural learning. Once we obtain an estimate $\hat{\theta}$ of the structural parameter, we can use the learning algorithm $\mathcal{A}_\ell(S_\ell, \hat{\theta})$ to obtain predictor $\hat{f}_{\ell, \theta}$ for each ℓ .

As an example, assume that we estimate the accuracy of $\hat{f}_{\ell, \theta}$ using a validation set $T_\ell = \{(\bar{\mathbf{X}}_j^\ell, \bar{Y}_j^\ell)\}_{j=1, \dots, \bar{n}_\ell}$, then we may simply let $\mathcal{O}_\ell(S_\ell, T_\ell, \theta) = \alpha_\ell \sum_{j=1}^{\bar{n}_\ell} L(\hat{f}_{\ell, \theta}(\bar{\mathbf{X}}_j^\ell), \bar{Y}_j^\ell)$, where $\alpha_\ell > 0$ are weighting parameters. It is also possible to estimate the accuracy of the learned predictor based on the training set alone using the standard learning theory for empirical risk minimization. This approach will be employed in Section 3, and leads to practical algorithms that can be formulated as optimization problems.

3. Algorithms

In this section, we develop a specific learning algorithm under the standard machine learning framework. The basis of our learner is *joint empirical risk minimization*, which will be analyzed in Appendix A. We consider linear prediction models since they have been shown to be effective in many practical applications. These methods include state-of-the-art machine learning algorithms such as kernel machines and boosting.

3.1 Joint Empirical Risk Minimization

Based on the framework outlined in Section 2, we are interested in finding a hypothesis space $\mathcal{H}_{\ell, \theta}$, using an estimator of the form (2). As being pointed out in Section 2, conceptually this could be achieved using a validation set. However, such an approach can lead to a quite difficult computational procedure since we have to optimize the empirical risk on the training data for each possible value of θ , and then choose an optimal θ on the validation set. Therefore for complicated structures with continuous θ parameter such as the model we consider in Section 3, this approach is not feasible.

A more natural method is to perform a joint optimization on the training set, with respect to both the predictors $\{f_\ell\}$, and the structural parameter θ . To this end, we will

consider the model given by equation (1), and pose it as a joint optimization problem over the m problems, where θ is the shared structural parameter:

$$[\hat{\theta}, \{\hat{f}_\ell\}] = \arg \min_{\theta \in \Gamma, \{f_\ell \in \mathcal{H}_{\ell, \theta}\}} \sum_{\ell=1}^m \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} L(f_\ell(\mathbf{X}_i^\ell), Y_i^\ell). \quad (3)$$

Since the shared structural parameter θ depends on m problems, it can be more reliably estimated by joint minimization. For completeness, we include a theoretical analysis in Appendix A.

3.2 Structural Learning with Linear Predictors

In order to derive a practical algorithm from (3), we shall consider a specific joint model which can be solved numerically. Specifically, we employ linear prediction models for the multiple tasks, and assume that the underlying structure is a shared low-dimensional subspace. Although not necessarily most general, this model leads to a simple and intuitive computational procedure. As we shall also see later, it is quite effective for semi-supervised learning problems that we are interested in.

Given the input space \mathcal{X} , a linear predictor is not necessarily linear on the original space, but rather can be regarded as a linear functional on a high dimensional feature space \mathcal{F} . We assume there is a known feature map $\Phi : \mathcal{X} \rightarrow \mathcal{F}$. A linear predictor f is determined by a weight vector \mathbf{w} : $f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x})$. In order to apply the structural learning framework, we consider a parameterized family of feature maps. In this setting, the goal of structural learning may be regarded as learning a good feature map. For the specific formulation which we consider in this paper, we assume that the overall feature map contains two components: one component is with a known high-dimensional feature map, and the other component is a parameterized low-dimensional feature map. That is, the linear predictor has a form

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + \mathbf{v}^T \Psi_\theta(\mathbf{x}),$$

where \mathbf{w} and \mathbf{v} are weight vectors specific for each prediction problem, and θ is the common structure parameter shared by all problems.

In order to simplify numerical computation, we further consider a simple linear form of feature map, where $\theta = \Theta$ is an $h \times p$ dimensional matrix, and $\Psi_\theta(\mathbf{x}) = \Theta \Psi(\mathbf{x})$, with Ψ a known p -dimensional vector function. We now can write the linear predictor as:

$$f_\Theta(\mathbf{w}, \mathbf{v}; \mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + \mathbf{v}^T \Theta \Psi(\mathbf{x}).$$

This hypothesis space (with appropriate regularization conditions) is analyzed in Appendix A after Theorem 4. We point out there that the key idea of this formulation is to discover a shared low-dimensional predictive structure parameterized by Θ .

Applying (2) with $\mathcal{O}(S_\ell, T_\ell, \theta)$ given by regularized empirical risk, we obtain the following formulation:

$$[\{\hat{\mathbf{w}}_\ell, \hat{\mathbf{v}}_\ell\}, \hat{\Theta}] = \arg \min_{\{\mathbf{w}_\ell, \mathbf{v}_\ell\}, \Theta} \left[r(\Theta) + \sum_{\ell=1}^m \left(g(\mathbf{w}_\ell, \mathbf{v}_\ell) + \frac{1}{n_\ell} \sum_{i=1}^{n_\ell} L(f_\Theta(\mathbf{w}_\ell, \mathbf{v}_\ell; \mathbf{X}_i^\ell), Y_i^\ell) \right) \right], \quad (4)$$

where $g(\mathbf{w}, \mathbf{v})$ is an appropriate regularization condition on the weight vector (\mathbf{w}, \mathbf{v}) , and $r(\Theta)$ is an appropriate regularization condition on the structural parameter Θ . In this formulation, we weight each problem equally (by dividing the number of instances n_ℓ) so that no problem will dominate the others. One may also choose other weighting schemes. Note that the regularized ERM method in (4) has the same form as (3). The main difference is that we replaced the hard-constrained regularization (picking the predictors from a hypothesis space) by its computationally more convenient version of penalized regularization. Up to appropriately defined Lagrangian multipliers, these two formulations are equivalent.

If we consider kernel learning, and assume that the feature map $\Phi(\mathbf{x})$ belongs to a reproducing kernel Hilbert space, then equation (4) can be kernelized. There are several ways to do so. One possibility is to kernelize in the \mathbf{w} parameter — we simply replace the vector parameter \mathbf{w}_ℓ by n_ℓ dual parameters α_j^ℓ ($j = 1, \dots, n_\ell$), and the linear score $\mathbf{w}_\ell^T \Phi(\mathbf{X}_i^\ell)$ by $\sum_{j=1}^{n_\ell} \alpha_j^\ell K(\mathbf{X}_j^\ell, \mathbf{X}_i^\ell)$. For simplicity, we do not consider kernel methods in this paper.

3.3 Alternating Structure Optimization

It is possible to solve (4) using general purpose optimization methods. However, in this section, we show that by exploring the special structure of the formulation, we can develop a more interesting and conceptually appealing computational procedure. In general, we should pick L and g such that the formulation is convex for fixed Θ . However, the joint optimization over $\{\mathbf{w}_\ell, \mathbf{v}_\ell\}$ and Θ will become non-convex. Therefore, one typically can only find a local minimum with respect to Θ . This usually doesn't lead to serious problems since given the local optimal structural parameter Θ , the solution $\{\mathbf{w}_\ell, \mathbf{v}_\ell\}$ will still be globally optimal for every ℓ . Moreover, the algorithm which we propose later in section uses SVD for dimension reduction. At the conceptual level, the possible local optimality of Θ is not a major issue simply because the SVD procedure itself is already good at finding globally optimal low dimensional structure.

With fixed Θ , the computation of $\{\mathbf{w}_\ell, \mathbf{v}_\ell\}$ for each problem ℓ becomes decoupled, and various optimization algorithms can be applied for this purpose. The specific choice of such algorithms is not important for the purpose of this paper. In our experiments, for convenience and simplicity, we employ stochastic gradient descent (SGD), widely used in the neural networks literature. It was recently argued that this simple method can also work well for large scale convex learning formulations (Zhang, 2004).

In the following, we consider a special case of (4) which has a simple iterative SVD solution. Let $\Phi(\mathbf{x}) = \Psi(\mathbf{x}) = \mathbf{x} \in R^p$ with square regularization of weight vectors. Then we have

$$\begin{aligned} \{ \{\hat{\mathbf{w}}_\ell, \hat{\mathbf{v}}_\ell\}, \hat{\Theta} \} &= \arg \min_{\{\mathbf{w}_\ell, \mathbf{v}_\ell\}, \Theta} \sum_{\ell=1}^m \left(\frac{1}{n_\ell} \sum_{i=1}^{n_\ell} L((\mathbf{w}_\ell + \Theta^T \mathbf{v}_\ell)^T \mathbf{X}_i^\ell, Y_i^\ell) + \lambda_\ell \|\mathbf{w}_\ell\|_2^2 \right), \\ \text{s.t. } \Theta \Theta^T &= I_{h \times h}, \end{aligned} \quad (5)$$

with given constants $\{\lambda_\ell\}$. Note that in this formulation, the regularization condition $r(\Theta)$ in (4) is absorbed into the orthonormal constraint $\Theta \Theta^T = I_{h \times h}$, and thus does not need to be explicitly included.

In order to solve this optimization problem, we may introduce an auxiliary variable \mathbf{u}_ℓ for each problem ℓ such that $\mathbf{u}_\ell = \mathbf{w}_\ell + \Theta^T \mathbf{v}_\ell$. Therefore we may eliminate \mathbf{w} using \mathbf{u} to obtain:

$$[\{\hat{\mathbf{u}}_\ell, \hat{\mathbf{v}}_\ell\}, \hat{\Theta}] = \arg \min_{\{\mathbf{u}_\ell, \mathbf{v}_\ell\}, \Theta} \sum_{\ell=1}^m \left(\frac{1}{n_\ell} \sum_{i=1}^{n_\ell} L(\mathbf{u}_\ell^T \mathbf{X}_i^\ell, Y_i^\ell) + \lambda_\ell \|\mathbf{u}_\ell - \Theta^T \mathbf{v}_\ell\|_2^2 \right), \quad (6)$$

s.t. $\Theta \Theta^T = I_{h \times h}$.

At the optimal solution, we let $\hat{\mathbf{w}}_\ell = \hat{\mathbf{u}}_\ell - \hat{\Theta}^T \hat{\mathbf{v}}_\ell$.

In order to solve (6), we use the following alternating optimization procedure:

- Fix (Θ, \mathbf{v}) , and optimize (6) with respect to \mathbf{u} .
- Fix \mathbf{u} , and optimize (6) with respect to (Θ, \mathbf{v}) .
- Iterate until convergence.

One may also propose other alternating optimization procedures. For example, in the first step, we may fix Θ and optimize with respect to (\mathbf{u}, \mathbf{v}) .

In the alternating optimization procedure outlined above, with a convex choice of L , the first step becomes a convex optimization problem. There are many well-established methods for solving it (as mentioned earlier, we use SGD for its simplicity). We shall focus on the second step, which is crucial for the derivation of our method. It is easy to see that the optimization of (6) with fixed $\{\mathbf{u}_\ell\} = \{\hat{\mathbf{u}}_\ell\}$ is equivalent to the following problem:

$$[\{\hat{\mathbf{v}}_\ell\}, \hat{\Theta}] = \arg \min_{\{\mathbf{v}_\ell\}, \Theta} \sum_{\ell} \lambda_\ell \|\hat{\mathbf{u}}_\ell - \Theta^T \mathbf{v}_\ell\|_2^2, \quad \text{s.t. } \Theta \Theta^T = I_{h \times h}. \quad (7)$$

Using simple linear algebra, we know that with fixed Θ ,

$$\min_{\mathbf{v}_\ell} \|\hat{\mathbf{u}}_\ell - \Theta^T \mathbf{v}_\ell\|_2^2 = \|\hat{\mathbf{u}}_\ell\|_2^2 - \|\Theta \hat{\mathbf{u}}_\ell\|_2^2,$$

and the optimal value is achieved at $\hat{\mathbf{v}}_\ell = \Theta \hat{\mathbf{u}}_\ell$. Now by eliminating \mathbf{v}_ℓ and use the above equality, we can rewrite (7) as

$$\hat{\Theta} = \arg \max_{\Theta} \sum_{\ell=1}^m \lambda_\ell \|\Theta \hat{\mathbf{u}}_\ell\|_2^2, \quad \text{s.t. } \Theta \Theta^T = I_{h \times h}.$$

Let $\mathbf{U} = [\sqrt{\lambda_1} \hat{\mathbf{u}}_1, \dots, \sqrt{\lambda_m} \hat{\mathbf{u}}_m]$ be an $p \times m$ matrix, we have

$$\hat{\Theta} = \arg \max_{\Theta} \text{tr}(\Theta \mathbf{U} \mathbf{U}^T \Theta^T), \quad \text{s.t. } \Theta \Theta^T = I_{h \times h},$$

where $\text{tr}(A)$ is the trace of matrix A . It is well-known that the solution of this problem is given by the SVD (singular value decomposition) of \mathbf{U} : let $\mathbf{U} = V_1 D V_2^T$ be the SVD of \mathbf{U} (assume that the diagonal elements of D are arranged in decreasing order), then the rows of $\hat{\Theta}$ are given by the first h rows of V_1^T (left singular vectors corresponding to the largest h singular values of \mathbf{U}). We now summarize the above derivation into an algorithm described in Figure 2, which solves (5) by alternating optimization of \mathbf{u} and (Θ, \mathbf{v}) .

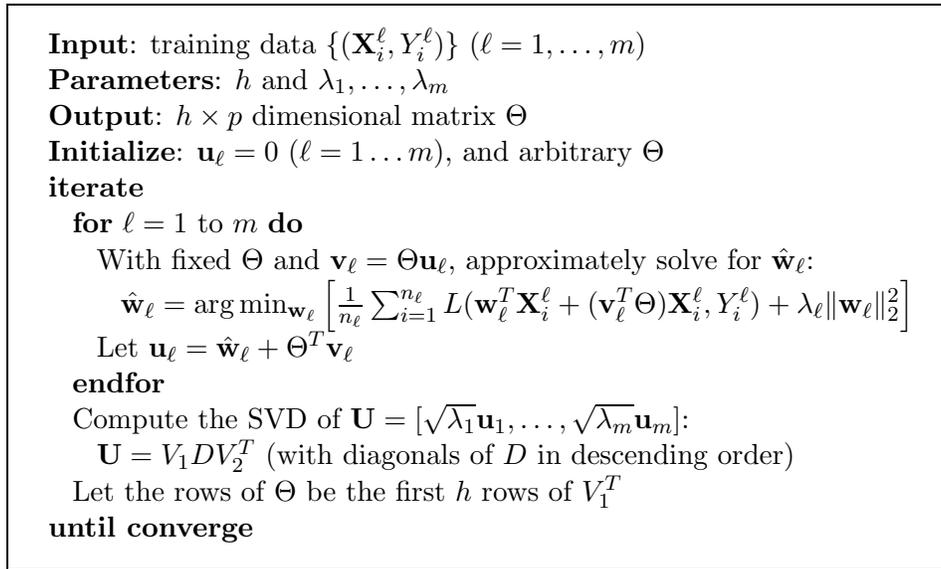


Figure 2: SVD-based Alternating Structure Optimization Algorithm

Note that since the objective value in (5) decreases at each iteration, the procedure produces parameters $\{\hat{\mathbf{w}}_\ell, \hat{\mathbf{v}}_\ell\}, \hat{\Theta}$ with converging objective values. In general, the parameters $\{\hat{\mathbf{w}}_\ell, \hat{\mathbf{v}}_\ell\}, \hat{\Theta}$ will also converge (to a local optimal solution). However, in reality, it is usually sufficient to use the Θ parameter from the first iteration of the procedure. This is because the performance of our model is not very sensitive to a small perturbation of the structural parameter Θ . The main dimensional reduction effect is already well captured by SVD in the first iteration.

It is important to point out that our SVD-based alternating structure optimization (SVD-ASO) procedure is fundamentally different from the usual principal component analysis (PCA) which can be regarded as dimension reduction in the data space \mathcal{X} . However, the dimension reduction performed in the SVD-ASO algorithm is on the predictor (classifier) space instead of the data space. This is possible because we observe multiple predictors from multiple learning tasks. If we regard the observed predictors as sample points of the predictor distribution in the predictor space (corrupted with estimation error, or noise), then our algorithm can be interpreted as finding the “principal components” of these predictors. Consequently the method directly looks for low-dimensional structures with the highest predictive power. By contrast, the principal components of input data in the data space do not necessarily have good predictive power.

3.4 An Extension of the Basic SVD-ASO Algorithm

One may extend (5) and the SVD-ASO procedure in various ways. For example, if \mathbf{x} belongs to an infinite dimensional Hilbert space, then the SVD in Figure 2 can be replaced by the corresponding kernel principal component analysis. However, this generalization is outside the scope of the analysis given in the paper.

In our experiments, we use another extension, where features (components of \mathbf{x}) are grouped into different types and the SVD dimension reduction is computed separately for each group. This is important since in applications, features are not homogeneous. If we know that some features are more similar to each other (e.g. they have the same type), then it is reasonable to perform a more localized dimension reduction among these similar features. To formulate this idea, we divide input features into G groups, and rewrite each input data-point \mathbf{X}_i^ℓ as $[\mathbf{X}_{i,t}^\ell]_{t=1,\dots,G}$, where t is the feature type which specifies which group the feature is in. Each $\mathbf{X}_{i,t}^\ell \in R^{p_t}$, and thus $\mathbf{X}_i^\ell \in R^p$ with $p = \sum_{t=1}^G p_t$. We associate each group t with a structural parameter $\Theta_t \in R^{h_t \times p_t}$, which is a projection operator of this feature type into h_t dimensional space. Equation (5) can be replaced by the following structural learning method:

$$\begin{aligned}
 [\{\hat{\mathbf{w}}_{\ell,t}, \hat{\mathbf{v}}_{\ell,t}\}, \{\hat{\Theta}_t\}] = & \arg \min_{\{\mathbf{w}_{\ell,t}, \mathbf{v}_{\ell,t}\}, \{\Theta_t\}} \sum_{\ell=1}^m \left(\frac{1}{n_\ell} \sum_{i=1}^{n_\ell} L \left(\sum_{t=1}^G (\mathbf{w}_{\ell,t} + \Theta_t^T \mathbf{v}_{\ell,t})^T \mathbf{X}_{i,t}^\ell, Y_i^\ell \right) \right. \\
 & \left. + \sum_{t=1}^G \lambda_{\ell,t} \|\mathbf{w}_{\ell,t}\|_2^2 \right), \tag{8} \\
 \text{s.t. } \forall t \in \{1, \dots, G\} : & \quad \Theta_t \Theta_t^T = I_{h_t \times h_t}.
 \end{aligned}$$

Similarly as before, we can introduce auxiliary variables $\mathbf{u}_{\ell,t} = \mathbf{w}_{\ell,t} + \Theta_t^T \mathbf{v}_{\ell,t}$, and perform alternating optimization over \mathbf{u} and (Θ, \mathbf{v}) . The resulting algorithm is essentially the same as the SVD-ASO method in Figure 2, but with the SVD dimension reduction step performed separately for each feature group t .

Some other extensions of the basic algorithm can also be useful for certain applications. For example, we may choose to regularize only those components of \mathbf{w}_ℓ which correspond to the non-negative part of \mathbf{u}_ℓ (we may still regularize the negative part of \mathbf{u}_ℓ , but using the corresponding components of \mathbf{u}_ℓ instead of \mathbf{w}_ℓ). The reason for doing so is that the positive weights of a linear classifier are usually directly related to the target concept, while the negative components often yield much less specific information. The resulting method can be easily formulated and solved by a variant of the basic SVD-ASO algorithm. In effect, in the SVD computation, we only use the positive components of \mathbf{u}_ℓ .

4. Semi-Supervised Learning

We are now ready to illustrate how to apply the structural learning paradigm developed earlier in the paper to the semi-supervised learning setting. The basic idea is to create auxiliary problems using unlabeled data, and then employ structural learning to reveal predictive structures intrinsic to the underlying input space \mathcal{X} .

4.1 Learning from Unlabeled Data Through Structural Learning

We systematically create multiple prediction problems from unlabeled data. We call these *created* prediction problems *auxiliary problems*, while we call the original supervised prediction problem (which we are interested in) the *target problem*.

Our method consists of the following two steps:

1. Learn a good structural parameter θ by performing a joint empirical risk minimization on the auxiliary problems, using originally unlabeled data that are automatically ‘labeled’ with auxiliary class labels.
2. Learn a predictor for the target problem by empirical risk minimization on the originally labeled data, using θ computed in the first step. In particular, in our bi-linear formulation (Section 3), we fix Θ and optimize (8) with respect to \mathbf{w} and \mathbf{v} for the target problem.

The first step seeks a hypothesis space \mathcal{H}_θ through learning the predictive functional structure shared by auxiliary predictors. If auxiliary problems are, to some degree, related to the target task, then the obtained hypothesis space \mathcal{H}_θ , which improves the average performance of auxiliary predictors, will also help the target problem. Therefore, the relevancy of auxiliary problems plays an important role in our method. We will return to this issue in the next section.

An alternative to the above two-step procedure is to perform a joint empirical risk minimization on the target problem (with labeled data) and on the auxiliary problems (with unlabeled data) at once. However, in our intended applications, the number of labeled data available for the target problem is usually small. Therefore the inclusion of the target predictor in the joint ERM will not have a significant impact.

4.2 Auxiliary Problem Creation

Our approach to semi-supervised learning requires auxiliary problems with the following characteristics:

- *Automatic labeling*: we need to automatically generate various “labeled” data for the auxiliary problems from unlabeled data.
- *Relevancy*: auxiliary problems should be related to the target problem (that is, they share a certain predictive structure) to some degree.

We consider two strategies for automatic generation of auxiliary problems: one in a completely unsupervised fashion, and the other in a partially supervised fashion. Some of the example auxiliary problems introduced in this section are used in our experiments described in Section 5.

We have briefly discussed the relationship of PCA and SVD-ASO in Section 3. In the above mentioned framework of semi-supervised learning, the standard PCA (applied to unlabeled data) can also be roughly regarded as a result of generating k auxiliary problems from k unlabeled data points so that the i -th problem has only one positive example (the i -th data point). In general, the strategies which we will suggest below are more flexible and more effective.

For clarity, we introduce the following two mini target tasks as running examples.

Text genre categorization Consider the task of assigning one of the three categories in $\{\text{SCIENCE, SPORTS, ECONOMY}\}$ to text documents. For this problem, suppose that we use frequencies of content words as features.

Word tagging Consider the task of assigning one of the three part-of-speech tags { NOUN, VERB, OTHER } to words in English sentences. For instance, the word “test” in “... a test procedure ...” should be assigned the tag NOUN, and that in “We will test it ...” should be assigned the tag VERB. For this problem, suppose that we use strings of the current and surrounding words as features.

4.2.1 UNSUPERVISED-STRATEGY: PREDICTING OBSERVABLE SUB-STRUCTURES

In the first strategy, we regard some observable substructures of the input data \mathcal{X} as auxiliary class labels, and try to predict these labels using other parts of the input data. For instance, for the word tagging task mentioned above, at each word position, we can create auxiliary problems by regarding the current word as auxiliary labels, which we want to predict using the surrounding words. We create one binary classification problem for each possible word value, and hence can obtain many auxiliary problems using this idea.

More generally, if we have a feature representation of the input data, then we may mask some features as unobserved, and learn classifiers to predict these ‘masked’ features (or some functional mapping of the masked features, e.g., bi-grams of left and current words) based on other features that are not masked. In the actual implementation, we just replace the masked feature values by zero, which has the same effect.

The automatic-labeling requirement is satisfied since the auxiliary labels are observable to us. To see why this technique may naturally meet the relevancy requirement, we note that feature components that can predict a certain masked feature are correlated to the masked feature, and thus are correlated among themselves. Therefore this technique helps us to identify correlated features with predictive power.

However, for optimal performance, it is clear that we should choose to mask (and predict) features that have good correlation to the target classes as auxiliary labels. The creation of auxiliary problems following this strategy is thus as easy or hard as designing features for usual supervised learning tasks. We can often make an educated guess based on task-specific knowledge. A wrong guess would result in adding some irrelevant features (originating from irrelevant θ -components), but it would not hurt ERM learners severely. On the other hand, potential gains from a right guess can be significant. Also note that given the abundance of unlabeled data, we have a wider range of choices than standard feature engineering in the supervised setting. For example, high-order features that suffer from the data sparseness problem in the supervised setting may be used in auxiliary problems due to the vast amount of unlabeled data that can provide more reliable statistics. The low-dimensional predictive structure discovered from the high-order features can then be used in the supervised task without causing the data-sparseness problem. This is because the rare features will be properly combined in the projection matrix Θ , so that the combined low-dimension directions will appear more frequently (and more correlated to the class-label). The example provided in Section 4.3 demonstrates this point.

The following examples illustrate auxiliary problems potentially useful for our example mini tasks.

Ex 1. Predict frequent words for text genre categorization. It is intuitive that content words that occur frequently in a document are often good indicators of the genre of that document. Let us split content words into two sets W_1 and W_2 (after removing

appropriate stop words). An auxiliary task we define is as follows. Given document x , predict the word that occurs most frequently in x , among the words in set W_1 . The learner only uses the words in W_2 for this prediction. This task breaks down to $|W_1|$ binary prediction problems, one for each content word in W_1 .¹

For example, let

$$\begin{aligned} W_1 &= \{\text{“stadium”, “scientist”, “stock”}\}, \\ W_2 &= \{\text{“baseball”, “basketball”, “physics”, “market”}\}. \end{aligned}$$

We treat members of W_1 as unobserved, and learn to predict whether the word “stadium” occurs more frequently in a document than “scientist” and “stock” by observing the occurrences of “baseball”, “basketball”, “physics”, and “market”. Similarly, the second problem is to predict whether “scientist” is more frequent than “baseball” and “stock”. Essentially, through this auxiliary problem, we learn the textual context in W_2 that implies that the word “stadium” occurs frequently in W_1 . Assuming that “stadium” is a strong indicator of SPORTS, the problem indirectly helps to learn the correlation of W_2 members to the target class SPORTS from unlabeled data.

Ex 2. Predict word strings for word tagging. As we have already discussed above, an example auxiliary task for word tagging is to predict the word string at the current position by observing the corresponding words on the left and the right. Using this idea, we can obtain $|W|$ binary prediction problems where W is a set of all possible word strings. Another example is to predict the word on the left by observing the words at the current and right positions. The underlying assumption is that word strings (at the current and left positions) have strong correlations to the target problem – whether a word is NOUN or VERB.

4.2.2 PARTIALLY SUPERVISED-STRATEGY: PREDICTING THE BEHAVIOR OF TARGET CLASSIFIER

The second strategy is motivated by co-training. We use two (or more) distinct feature maps: $\Phi_1 : \mathcal{X} \rightarrow \mathcal{F}$ and $\Phi_2 : \mathcal{X} \rightarrow \mathcal{F}$. First, we train a classifier for the target task, using the feature map Φ_1 and the labeled data. The auxiliary tasks are to predict the behavior of this classifier (such as predicted labels, assigned confidence values, and so forth), by using the other feature map Φ_2 . Note that unlike co-training, we only use the classifier as a means of creating auxiliary problems that meet the relevancy requirement, instead of using it to bootstrap labels. The semi-supervised learning procedure following this strategy is summarized as follows.

1. Train a classifier T_1 with labeled data Z for the target task, using feature map Φ_1 .
2. Generate labeled data for auxiliary problems by applying T_1 to unlabeled data.
3. Learn structural parameter θ by performing joint ERM on the auxiliary problems, using only the feature map Φ_2 .

1. One may also consider variations of this idea, such as predicting whether a content word in W_1 appears more often than a certain threshold, or in the top- k most frequent list of x .

4. Train a final classifier with labeled data Z , using θ computed above and some appropriate feature map Ψ .

Ex 3. Predict the prediction of classifier T_1 . The simplest auxiliary task created by following this strategy is the prediction of the class labels proposed by classifier T_1 . When the target task is c -way classification, c binary classification problems are obtained in this manner. For example, suppose that we train a classifier using only one half of content words for the text genre categorization task. Then, one of auxiliary problems will be to predict whether this classifier would propose SPORTS category or not, based on the other half of content words only.

Ex 4. Predict the top- k choices of the classifier. Another example is to predict the combinations of k (a few) classes to which T_1 assigns the highest confidence values. Through this problem, fine-grained distinctions (related to intrinsic sub-classes of target classes) can be learned. From a c -way classification problem, $c!/(c-k)!$ binary prediction problems can be created. For instance, predict whether T_1 assigns the highest confidence values to SPORTS and ECONOMY in this order.

Ex 5. Predict the range of confidence values produced by the classifier. Yet another example is to predict the proposed labels in conjunction with the range of confidence values. For instance, predict whether T_1 would propose SPORTS with confidence greater than 0.5.

4.3 Discussions

We have introduced two strategies for creating auxiliary problems in this section. One is unsupervised, and the other partially supervised.

For the unsupervised strategy, we try to predict some sub-structures of the input using some other parts of the input. This idea is related to the discussion in Section 2.3, where we have argued that there are often good structures (or smoothness conditions) intrinsic to the input space. These structures can be discovered from auxiliary problems. For text data, some words or linguistic usages will have similar meanings. The smoothness condition is related to the fact that interesting predictors for text data (often associated with some underlying semantic meanings) will take similar values when a linguistic usage is substituted by one that is closely related. This smoothness structure can be discovered using structural learning, and specifically by the method we proposed in Section 3. In this case, the space of smooth predictors corresponds to the most predictive low dimensional directions which we may discover using the SVD-ASO algorithm. An example of computed Θ is given in Section 5.2.8, which supports the argument. It is also easy to see that this reasoning is not specific to text. Therefore the idea can be applied to other data such as images. In the following, we will briefly explain the underlying intuition on why the un-supervised auxiliary problems we create are helpful for the supervised task, and leave the development of a more rigorous and general theory to future investigation.

Suppose we split the features into two parts W_1 and W_2 , and then predict W_1 based on W_2 . Suppose features in W_1 are correlated to the class labels (but not necessarily correlated among themselves). Then, the auxiliary prediction problems are related to the target task, and thus can reveal useful structures of W_2 . Under appropriate conditions, features in W_2

with similar predictive performance tend to map to similar low-dimensional vectors through Θ . This effect can be empirically observed in Section 5.2.8. We shall only use a simple but concrete example to illustrate the main idea. Assume that words are divided into five disjoint sets $T_j : -2 \leq j \leq 2$, with binary label $y \in \{-1, 1\}$. Assume also for simplicity that every document contains only two words x_1 and x_2 , where $x_1 \in W_1 = \cup_{j=-1,0,1} T_j$, and $x_2 \in W_2 = \cup_{j=-2,2} T_j$. Assume further that given class label $y = \pm 1$, x_1 and x_2 are independent, where x_1 is uniformly distributed over $T_0 \cup T_y$ and x_2 is uniformly distributed over T_{2y} . Then by predicting $x_1 = \ell$ based on x_2 with least squares, we obtain for each $y \in \{\pm 1\}$, identical weight-components for all words in T_{2y} (due to the exchangeability of words in each T_{2y}). Thus after dimension reduction, only two rows of Θ have non-zero singular values. The Θ -feature reduces all words in T_2 to a single vector in R^2 , and all words in T_{-2} into another single vector in R^2 . This gives a helpful grouping effect (words with similar predictive performance are grouped together). It is clear that in this example, we gain predictive ability by using unsupervised structure discovery. This is because the original word-spaces T_2 and T_{-2} may be extremely large, which means that one will not be able to learn very well from a small number of training examples (since each word does not occur often enough). By grouping all words in T_2 (also in T_{-2}) together, we obtain a feature that is completely correlated to the class label due to our data generation process. Therefore the grouping effect makes the originally hard problem much easier to learn. This example can be extended to a more general theory, which we shall leave to further exploration. The consequence of the example is observable in practice, as demonstrated in Section 5.2.8.

Although the above discussion implies that it is possible to find useful predictive structures even if we do not intentionally create problems to mimic the target problem, it is also clear that auxiliary problems more closely related to the target problem will be more beneficial. This is our motivation to propose the partially supervised strategy for creating auxiliary problems. Using this idea, it is always possible to create relevant auxiliary problems that are closely related to the supervised problem without knowing the effectiveness of the individual features. In practical applications, we observe that it can be desirable to create as many auxiliary problems as possible, as long as there is some reason to believe in their relevancy to the target task. This is because the risk is relatively minor, while the potential gain from a good structure is large.

Moreover, the auxiliary problems introduced above (and used in our experiments of the next section) are merely possible examples. One advantage of this approach to semi-supervised learning is that one may design a wide variety of auxiliary problems for learning various aspects of the target problem from unlabeled data. Structural learning provides a theoretical foundation and a general framework for carrying out possible new ideas.

5. Experiments

We study the performance of our structural learning-based semi-supervised method on text categorization, natural language tagging/chunking, and image classification tasks. The experimental results show that we are able to improve state-of-the-art supervised learning methods even for some problems with relatively large number of labeled data (e.g. 200K labeled data for named entity recognition).

5.1 Implementation

We experiment with the following semi-supervised learning procedure:

1. If required for auxiliary label generation, train classifiers T_i using labeled data Z and appropriate feature maps Ψ_i .
2. For all the auxiliary problems, assign auxiliary labels to unlabeled data.
3. Compute structure matrix Θ by performing the SVD-ASO procedure (Section 3) using all auxiliary problems on the data generated above. We use the extended version to take advantage of natural feature splits, and iterate once.
4. Fix Θ and obtain the final classifier by optimizing (8) with respect to \mathbf{w} and \mathbf{v} , using labeled data Z .

In all settings (including baseline methods), the loss function is a modification of the Huber’s robust loss for regression:

$$L(p, y) = \begin{cases} \max(0, 1 - py)^2 & \text{if } py \geq -1 \\ -4py & \text{otherwise} \end{cases},$$

with square regularization ($\lambda = 10^{-4}$). It is known that the modified Huber loss works well for classification, and has some advantages, although one may select other loss functions such as SVM or logistic regression. The specific choice is not important for the purpose of this paper. The training algorithm is stochastic gradient descent (SGD) as in (Zhang, 2004). We fix h_t (dimension of Θ_t) to 50, and use it for all the settings unless otherwise specified.

5.2 Text Categorization Experiments

We report text categorization performance on the 20-newsgroup corpus and the Reuters-RCV1 corpus (also known as “new Reuters”).

5.2.1 FEATURE REPRESENTATION

Our feature representation uses word frequencies after removing function words and common stopwords, and normalizes feature vectors into unit vectors.

5.2.2 AUXILIARY PROBLEMS FOR TEXT CATEGORIZATION

We experiment with the following types of auxiliary problems:

- **Freq**: predicts the most frequent word by observing one half of the words (as in Section 4.2.1. Ex 1).
- **Top- k** : predicts combinations of the top- k choices of the classifier trained with labeled data (as in Section 4.2.2. Ex 4).
- **Multi- k** : for the multi-category target task, predicts the top- k choices of the classifier (trained with labeled data), regarding them as multi-category auxiliary labels. The

number k is set to the average number of categories per instance, obtained from the labeled data.

Feature splits are randomly generated.

5.2.3 DATA

20-newsgroup corpus The 20-newsgroup corpus is one of the standard data sets for text categorization, which consists of 20K documents from 20 newsgroups, with 1K documents per group. The task is to classify documents into these 20 newsgroups ranging over a variety of topics – computer hardware, baseball, bikes, religions, middle east issues, and so on. In pre-processing, we removed the header lines (subjects, newsgroup names, senders, and so forth) from all documents. We held out 1K documents as the test set, and arbitrarily split the rest of the corpus into the training set (2K documents) and the unlabeled data set (17K documents).

Reuters-RCV1 corpus (new Reuters) From the Reuters-RCV1 corpus, we randomly generate disjoint sets of labeled (1K), unlabeled (20K), and test (3K) examples. The Reuters-RCV1 corpus differs from the 20-newsgroup corpus in several ways. The number of categories is 102, which is five times larger than that of the 20-newsgroup corpus; the categories are organized into three-level hierarchies; each document may be assigned multiple categories — about three categories per document on average. The Reuters-RCV1 corpus preserves the natural distribution of the categories whereas the 20-newsgroup corpus has a completely uniform distribution, generated by intentionally choosing the same number of documents from each newsgroup.

5.2.4 EVALUATION METRIC

To measure the performance of the final classifier on the test sets, for singly-labeled tasks, we choose one category that produces the highest confidence value (inner product) and report classification accuracy. For multiply-labeled tasks, we choose categories that produce positive confidence values, and report the micro-averaged F-measure.

5.2.5 TEXT CATEGORIZATION PERFORMANCE RESULTS

20-newsgroup results Figure 3 (a) shows the accuracy results on the 20-newsgroup data in comparison with the supervised setting as the baseline. We show the averaged results over 10 runs, each with labeled examples randomly drawn from the training set. The vertical bars are ‘one’ standard deviations. The symbol ‘semi’ stands for semi-supervised, followed by the types of auxiliary problems used. The semi-supervised methods obtain significant performance improvements (up to 22.2%) over the supervised method in all settings.

Reuters-RCV1 results Figure 3 (b) shows micro-averaged F-measure on the Reuters-RCV1 data in comparison to the supervised baseline. The performance trend is similar to that of the 20-newsgroup experiments. Significant performance improvements (up to 11.6%) over the supervised method are obtained in all settings.

Auxiliary problems: unsupervised vs. partially-supervised From the results in Figure 3, we observe that when a relatively small number of labeled data are used,

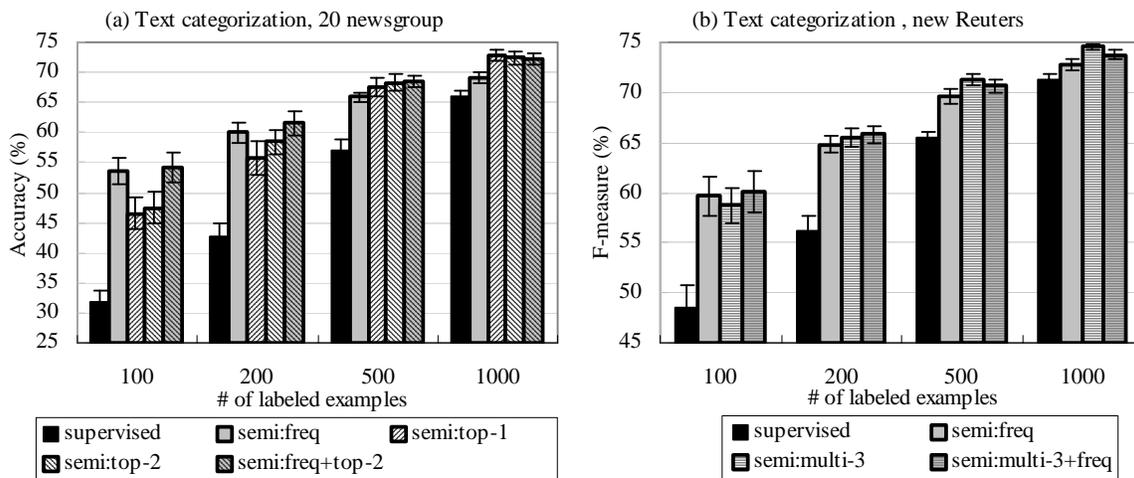


Figure 3: Text categorization performance results. Average over 10 runs. Vertical bars are standard deviations. (a) 20-newsgroup, (b) Reuters-RCV1.

freq (which uses auxiliary problems created in an unsupervised manner) outperforms top- k /multi- k (partially-supervised). However it underperforms top- k /multi- k when a relatively large number of labeled data are used. Since freq learns from unlabeled data in an unsupervised fashion, its effectiveness is insensitive to the number of labeled data. In contrast, top- k /multi- k can take advantage of information in the labeled data when there is a reasonable amount of them. The best performance is often achieved when both types of auxiliary problems are used.

5.2.6 PERFORMANCE COMPARISON WITH OTHER METHODS

As we have mentioned, the idea of using partially supervised auxiliary problems is motivated by co-training. Therefore we test co-training for comparison.

Co-training implementation Our implementation follows the original work (Blum and Mitchell, 1998), with the same feature splits as used in our auxiliary problems. Initial classifiers are trained with labeled instances drawn from the training sets. We maintain a pool of 10K unlabeled instances while refilling it by randomly choosing instances from the unlabeled set. The two classifiers propose labels for the unlabeled instances in this pool. For each classifier, we choose 1000 instances with high confidence while preserving the class distribution observed in the initial labeled data. This is done by choosing the class label with probabilities according to the distribution, and then the highest confident instance for that class label. The chosen instances are added to the pool of labeled data with their automatically proposed labels. The process repeats until the unlabeled instances are exhausted.

Comparison with co-training Figure 4 shows the best possible performance of co-training (the optimally stopped co-training procedure) averaged over 10 runs, with 100 and 200 labeled examples on the 20-newsgroup and the Reuters-RCV1 data. Our method

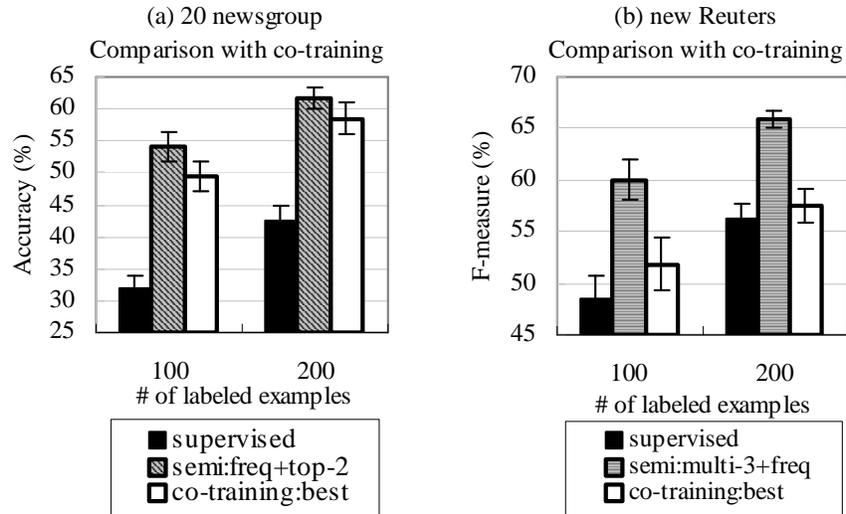


Figure 4: Comparison with co-training: averaged performance over 10 runs with standard deviations.

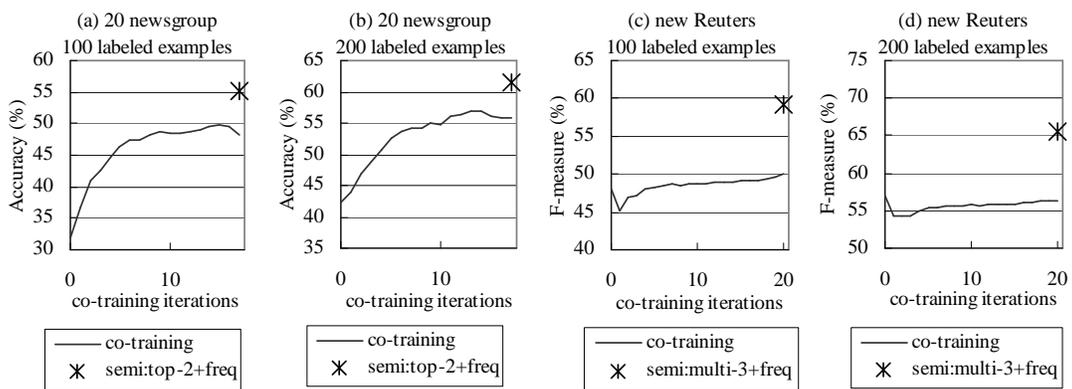


Figure 5: Co-training performance in typical runs, versus the number of iterations.

# of labeled examples	BN04 best (manifold)	ASO-semi
100	39.8	54.1
200	–	61.6
500	59.9	68.5
1000	64.0	72.3

Figure 6: Comparison with similar settings in BN04 (Belkin and Niyogi, 2004) on 20 newsgroup.

outperforms the best co-training performance in all of the four settings by up to 8.4%. In Figure 5, we plot the co-training performance versus co-training iterations in typical runs.

As shown in Figure 6, our results outperform BN04 (Belkin and Niyogi, 2004)’s manifold-based semi-supervised learning method. They are also consistent with NMTM00 (Nigam et al., 2000)’s EM results. Since NMTM00 didn’t report the exact numbers (we can only approximately read their results from a graph), we cannot include them in Figure 6. The performance of EM is usually similar to that of co-training as well as self-training frequently used in NLP. Although quite successful for the 20 newsgroup data, as we shall see later, co-training and self-training do not perform very well for more difficult tasks.

5.2.7 PERFORMANCE DEPENDENCY ON h

Recall that throughout the experiments, we fix the number of rows of Θ_t to a constant $h_t = 50$, as described in Section 5.1. Also recall that on text categorization, Θ_t is derived from auxiliary problems that use the t -th feature map. In this section, we study the performance dependency on the dimensionality h_t .

We are interested in the range of h_t roughly from 10 to 100. Figure 7 plots the performance on the 20-newsgroup and the Reuters-RCV1 corpora, in relation to $h_t = 5, 10, 15, 20, 30, \dots, 100$. The results show that the method is insensitive to the change of dimension h_t in a relatively large range. In practice, this is a significant advantage over other dimension reduction approaches, which are typically sensitive to the choice of dimensions, or bootstrapping approaches, which are often sensitive to parameter settings.

5.2.8 INTERPRETATION OF Θ

In order to gain some insights into the information obtained from unlabeled data, we show several significant entries of matrix Θ – the entries whose absolute values are:

- the largest in the columns (corresponding to features), and
- within the 100 largest among the positive (or negative) entries in the rows.

In the table below, we show at most ten entries chosen in this manner from the rows corresponding to the five most significant singular values. Θ was computed from the freq (unsupervised) auxiliary problems on the 20-newsgroup unlabeled data.

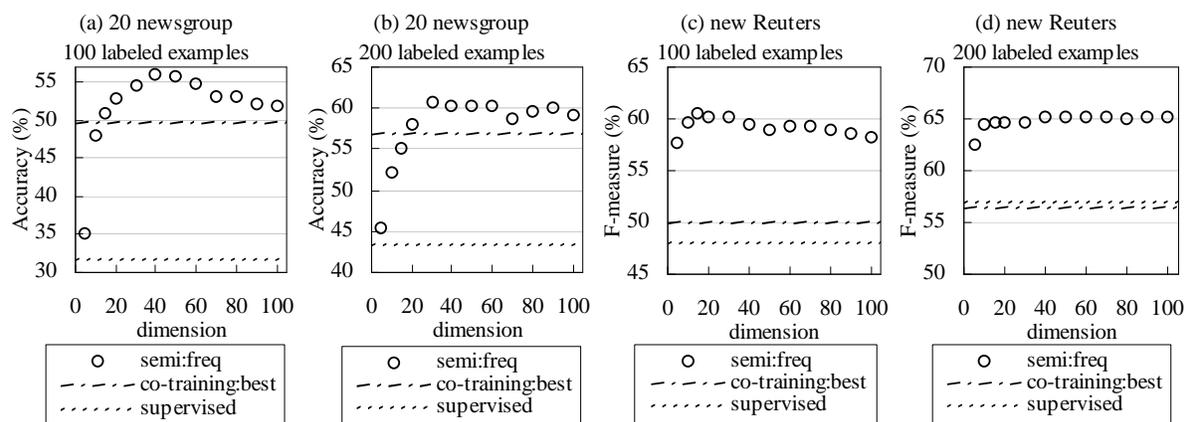


Figure 7: Performance dependency on h_t (the rank of Θ_t) in particular runs.

The second row appears to capture the distinctions between computers and religion. The third row distinguishes sports and the middle east issues. The positive entries of the fifth row appear to be about motor vehicles, and the negative entries are about printers. These topics are, indeed, relevant to the themes of the twenty newsgroups.

row#	features
2 +	pc, vesa, ibm, boards
-	god, christian, bible, exist, doctrine, nature, worship, athos.rutgers.edu
3 +	team, detroit, series, leafs, play, cup, playoffs, played, penguins, devils
-	israel, peace, jewish, lebanese, israelis, land, gaza, civilians, palestine, syria
4 +	files, jpeg, pov, utility, ms-windows, icon
-	eisa, nubus, agents, attorney
5 +	oil, bikes, front, brake, rear, transmission, owner, driving, dogs, highway
-	printer, hp, ink, appreciate, bj-200, toner, printing, bubblejet, laserjet, gcc

5.3 Named Entity Chunking Experiments

We report named entity chunking performance on the CoNLL'03 shared-task² corpora (English and German). We choose this task because the original intention of this shared task was to test the effectiveness of semi-supervised learning methods (such as label bootstrap or co-training), and hence a large number of unlabeled data were made available. However, it turned out that none of the top performing systems used unlabeled data. One possible reason may be that the number of labeled data is relatively large (>200K). We show that by contrast, through our structural-learning based semi-supervised learning, it is possible to obtain results better than any of the top systems, using unlabeled data as the only additional resource. In particular, we do not use gazetteer information, which was used in all other systems.

The CoNLL corpora are annotated with four types of named entities: persons, organizations, locations, and miscellaneous names (e.g., “World Cup”). As is commonly done, we en-

2. <http://cnts.uia.ac.be/conll2003/ner>.

code chunk information into word tags to cast the chunking problem to that of word tagging, and perform Viterbi-style decoding. We use the official training/development/test splits, as provided by the shared-task organizers. Our unlabeled data sets consist of 27 million words (English) and 35 million words (German), respectively. They were chosen from the same sources – Reuters and ECI Multilingual Text Corpus – as the training/development/test sets but disjoint from them.

5.3.1 FEATURE REPRESENTATION

Our feature representation is a slight modification of a simpler configuration reported in (Zhang and Johnson, 2003), which uses: token strings, parts-of-speech, character types, several characters at the beginning and the ending of the tokens, in a 5-token window around the current position; token strings in a 3-syntactic chunk window; labels of two tokens on the left to the current position; bi-grams of the current token and the label on the left; and the labels assigned to previous occurrences of the current word. These features are easily obtained without deep linguistic processing.

5.3.2 AUXILIARY PROBLEMS FOR NAMED ENTITY CHUNKING

We use four types of auxiliary problems and their combinations:

- Word prediction: predicts the word at the current (or left or right) position, using the features derived from the other tokens.
- Top-2: predicts the top-2 choices of the classifier. We split features into “left-context vs. the others” and “right-context vs. the others”. The rest is the same as Ex 4 in Section 4.2.2.

SVD is applied to each of the feature types separately. As for the word-prediction auxiliary problems, we only consider the instances whose current words are either nouns or adjectives since named entities mostly consist of these types. Also, we leave out all but 1000 auxiliary problems of each type that have the largest numbers of positive examples. This is to ensure that auxiliary predictors can be adequately learned from unlabeled data.

5.3.3 PERFORMANCE RESULTS ON THE CONLL ENGLISH/GERMAN CORPORA

Figures 8 (a) and (b) show the English F-measure results – (a) with small (10K-word) labeled data, and (b) with the entire training set (204K words). German results using the entire training set are shown in Figure 8 (c). Precision and recall results in the same settings are found in Figure 15.

Note that to facilitate comparisons with the supervised baselines, we do *not* use any gazetteers or any name lexicons. Thus, there are only two kinds of information sources: labeled data and unlabeled data. We confirm that performance improvements gained by unlabeled data are significant in all of the semi-supervised settings: up to 10.10% gains with small English labeled data, up to 3.86% with larger English labeled data, and up to 9.22% improvements on the German data.

We note that word-prediction (unsupervised) auxiliary problems are particularly effective when the number of labeled examples is relatively small or the training data differ

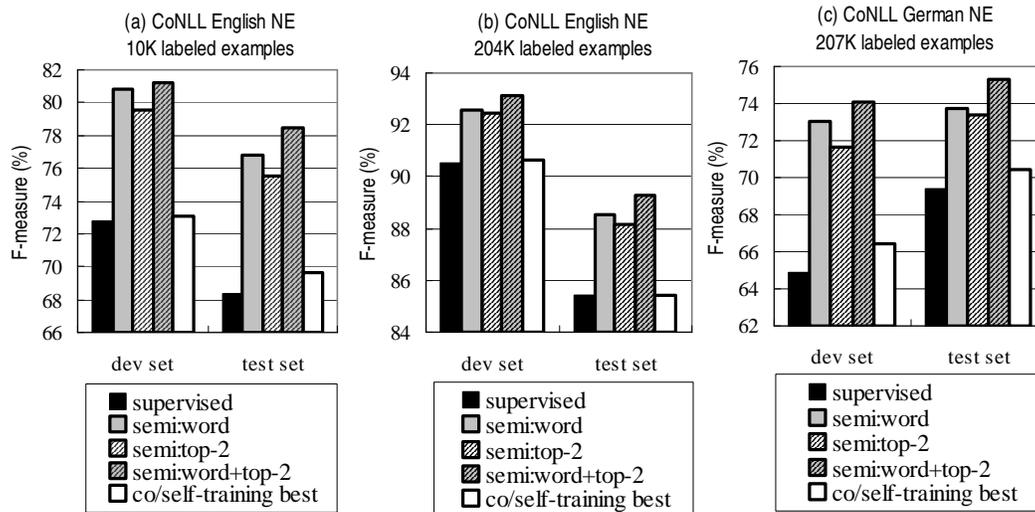


Figure 8: Named entity chunking F-measure performance. Without any gazetteer. For co- and self-training, the performance best among all the parameter settings (including the number of iterations) is shown. (a) CoNLL English corpus, 10K labeled examples. (b) CoNLL English corpus, 204K (the entire) labeled examples. (c) CoNLL German corpus, 207K (the entire) labeled examples.

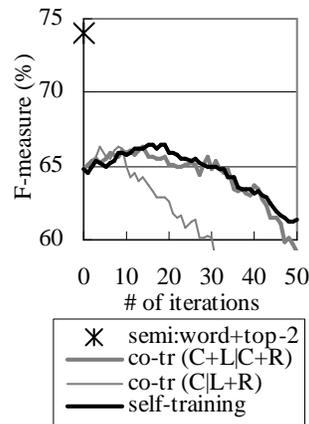


Figure 9: Co- and self-training named entity chunking performance in typical runs, versus the number of iterations. Tested on the German development set. In the legend, C, L, and R stand for current words, left context, and right context, respectively.

significantly from the test data. (The English test set is known to be less similar to the training set than the development set is, apparently because of the time periods from which the articles were drawn.) The best performance is achieved by combining all of the aux-

iliary problems. This performance trend is in line with that in the text categorization experiments.

Comparison with co- and self-training For comparison, we test co-training exploring parameter settings: pool size {50K,100K}, increment size {50, 100, 50K, 100K}, and commonly-used feature splits “current+left-context vs. current+right-context” and “current vs. context”. Single-view bootstrapping is sometimes called *self-training*. In addition, we test the basic self-training, which replaces multiple classifiers in the co-training procedure with a single classifier that employs all the features. The co- and self-training performance shown in Figures 8 and 15 is the best possible performance among all the parameter settings (including the number of iterations). Co- and self-training at their best improve recall but often degrades precision. Consequently, their F-measure improvements are relatively low, which demonstrates that it is not easy to benefit from unlabeled data on this task. Moreover, as shown in Figure 9, co- and self-training may rather degrade performance severely unless the iteration is optimally stopped. Such performance degradation (caused by contamination of automatically assigned labels) has also been observed in previous co-training studies on NLP tasks (e.g., Pierce and Cardie, 2001).

5.3.4 COMPARISON WITH PREVIOUS BEST RESULTS

English test set

System	F-measure	Additional resources
semi:word+top-2	<i>89.31</i>	<i>unlabeled data</i>
FIJZ03(Florian et al., 2003)	88.76	gazetteers; 1.7M-word labeled data
CN03(Chieu and Ng, 2003)	88.31	gazetteers (also very elaborated features)
KSNM03(Klein et al., 2003)	86.31	rule-based post processing

German test set

Systems	F-measure	Additional resources
semi:word+top-2	<i>75.27</i>	<i>unlabeled data</i>
FIJZ03	72.41	gazetteers
KSNM03	71.90	rule-based post processing
ZJ03(Zhang and Johnson, 2003)	71.27	gazetteers

Figure 10: Comparison with previous best results on CoNLL’03 shared task

In Figure 10, we compare our performance results with those of the previous top systems among the CoNLL’03 shared-task participants.

On both English and German data, we are able to achieve performance better than those of the top participants, although they used more elaborated features. We note that the previous best English results were achieved with the help of knowledge intensive resources – such as gazetteers provided by the organizer plus additional gazetteers of a large number of names (FIJZ03, CN03); and a large amount (1.7 million words) of labeled data annotated with finer-grained named entities (FIJZ03); and rule-based post processing (KSNM03). Recall that the study of semi-supervised learning is motivated by the potential unavailability

of such labor intensive resources. Hence, we feel that our results, which were obtained by using unlabeled data as the *only* additional resource, are very encouraging.

5.4 Part-of-Speech Tagging

We report part-of-speech (POS) tagging results on the Brown corpus. This corpus, annotated with 46 parts-of-speech, is one of the standard corpora for POS tagging research. We arbitrarily split the corpus into the labeled set (23K words), unlabeled set (1M words), and the test set (60K words).

The same auxiliary problems and feature representation (as in the named entity chunking experiments) are used, except for part-of-speech and syntactic chunk information. Following the convention, we use error rate to measure the performance. It can be seen from Figure 11 that over 20% error reductions are achieved by learning from unlabeled data.

supervised	8.9
semi:left+curr	7.0 (21.3%)
semi:top-1	6.9 (22.5%)
semi:left+curr+top-1	6.9 (22.5%)

Figure 11: Part-of-speech tagging error rates (%). The numbers in parentheses are error reduction ratio with respect to the supervised baseline.

5.5 Hand-Written Digit Image Classification

This experiment uses the MNIST data downloaded from <http://yann.lecun.com/exdb/mnist/>. It consists of a training set (60K examples) and a test set (10K examples) of 28-by-28 gray-scale hand-written digits. The task is to classify the image data into 10 digits, '0'–'9'.

We use a feature representation composed of location-sensitive bags of pixel blocks, similar to the bag-of-words model in text categorization. It consists of normalized counts of pixel blocks of various shapes in the four regions (top-left, top-right, bottom-right, bottom-left). (Normalization was done by scaling the vector for each shape/region into a unit vector.) The pixel blocks are black-white patterns of 16 pixels in the shape of: squares(4×4), rectangles(2×8 , 8×2), crossing lines (from top-left to bottom-right; from top-right to bottom-left), and dotted lines (horizontal and vertical). Using these features and trained with the entire training set (60K examples), the error rate in the supervised setting is 0.82%. This matches/surpasses state-of-the-art algorithms on the same data (reported on the MNIST data website) without additional image processing or transformation such as distortion or deskewing.

Auxiliary problems we used are partially-supervised. Feature splits were made by halving each image: features derived from top-left+top-right regions vs. those from bottom-left+bottom-right; top-left+bottom-left vs. top-right+bottom-right; and top-left+bottom-right vs. top-right+bottom-left.

In each run, labeled examples were randomly chosen from the training set, with the remaining training set used as unlabeled data. ASO-semi (Figure 12) consistently produced

significant performance improvements over the supervised baseline.³ It also outperforms a manifold-based semi-supervised learning method BN04 (Belkin and Niyogi, 2004) except when the number of labeled data is 100. The method in BN04 performs well for small labeled data. However, a disadvantage is that their method (which also requires dimension reduction) is more sensitive to the number of reduced dimensions. For example, with 100 labeled data, they achieved an error rate of 6.4 with 20 dimensions, but an error rate of 22.0 with 10 dimensions, and an error rate of 14.4 with 50 dimensions.

#labeled	supervised	ASO-semi	BN04 best (2nd best)
100	14.22 ± 2.90	9.13 ± 1.95	6.4 (14.4)
500	3.93 ± 0.22	3.05 ± 0.20	3.5 (3.6)
1000	2.83 ± 0.16	2.26 ± 0.11	3.2 (3.4)
5000	1.64 ± 0.07	1.47 ± 0.07	2.7 (2.9)

Figure 12: Error rates (%); average over 10 runs and standard deviation. MNIST hand-written digit image classification results on the test set. BN04 results (Belkin and Niyogi, 2004) are on the unlabeled portion of the training set.

(a) 20-newsgroup

# of labeled examples	100	200	500	1000
supervised	32.0	42.7	56.9	66.0
semi:freq	53.6 (+21.6)	60.0 (+17.3)	65.8 (+8.9)	69.1 (+3.1)
semi:top-1	46.6 (+14.6)	55.9 (+13.2)	67.6 (+10.7)	72.9 (+6.9)
semi:top-2	47.4 (+15.4)	58.4 (+15.7)	68.3 (+11.4)	<i>72.5</i> (+6.5)
semi:top-2+freq	<i>54.1</i> (+22.1)	<i>61.6</i> (+18.9)	<i>68.5</i> (+11.6)	72.3 (+6.3)

(b) Reuters-RCV1 corpus

# of labeled examples	100	200	500	1000
supervised	48.5	56.3	65.4	71.4
semi:freq	59.6 (+11.1)	64.8 (+8.5)	69.6 (+4.2)	72.8 (+1.4)
semi:multi-3	58.7 (+10.2)	65.5 (+9.2)	<i>71.2</i> (+5.8)	<i>74.6</i> (+3.2)
semi:multi-3+freq	<i>60.1</i> (+11.6)	<i>65.8</i> (+9.5)	70.7 (+5.3)	73.7 (+2.3)

Figure 13: Text categorization. Average over 10 runs. For each run, labeled examples were randomly drawn from the training set. (a) Accuracy on the 20-newsgroup corpus, (b) F-measure (micro-averaged) on Reuters-RCV1 corpus. Numbers in parentheses are performance improvements obtained from unlabeled data. The best performance in each column is italicized.

3. By contrast, co-training (with the same feature splits) sometimes rather degraded performance.

Data set	20-newsgroup		Reuters-RCV1	
# of labeled examples	100	200	100	200
co-training:highest	49.6 (+17.6)	58.5 (+15.8)	51.9 (+2.4)	57.4 (+1.1)
co-training:lowest	34.5 (+2.5)	45.5 (+2.8)	46.8 (-1.7)	53.9 (-2.4)

Figure 14: Co-training text categorization performance. The highest and lowest performance among the iterations, averaged over 10 runs. Accuracy on 20-newsgroup and micro-averaged F-measure on Reuters-RCV1 are shown. The numbers in parentheses are improvements over the supervised settings.

(a) English (10K labeled examples)

	development set			test set		
	prec.	recall	$F_{\beta=1}$	prec.	recall	$F_{\beta=1}$
supervised	72.04	73.46	72.74	70.52	66.25	68.32
co/self best	72.36	73.85	73.10 (+0.36)	71.12	68.20	69.63 (+1.31)
semi:word	<i>82.16</i>	79.45	80.78 (+8.04)	78.66	74.96	76.77 (+8.45)
semi:top-2	80.78	78.31	79.52 (+6.78)	77.60	73.58	75.54 (+7.22)
semi:word+top-2	82.06	<i>80.46</i>	<i>81.25 (+8.51)</i>	<i>79.91</i>	<i>76.98</i>	<i>78.42 (+10.10)</i>

(b) English (204K labeled examples)

	development set			test set		
	prec.	recall	$F_{\beta=1}$	prec.	recall	$F_{\beta=1}$
supervised	91.59	89.48	90.53	86.34	84.58	85.45
co/self best	91.63	89.68	90.64 (+0.11)	86.30	84.53	85.40 (-0.05)
semi:word	93.45	91.75	92.60 (+2.07)	89.04	88.05	88.54 (+3.09)
semi:top-2	93.05	91.89	92.46 (+1.93)	88.49	87.80	88.14 (+2.69)
semi:word+top2	<i>93.84</i>	<i>92.48</i>	<i>93.15 (+2.62)</i>	<i>89.54</i>	<i>89.09</i>	<i>89.31 (+3.86)</i>

(c) German (207K labeled examples)

	development set			test set		
	prec.	recall	$F_{\beta=1}$	prec.	recall	$F_{\beta=1}$
supervised	74.61	57.33	64.84	78.65	62.07	69.39
co/self best	72.02	61.72	66.47 (+1.63)	77.39	64.66	70.45 (+1.06)
semi:word	<i>82.04</i>	65.80	73.03 (+8.19)	82.23	66.78	73.71 (+4.32)
semi:top-2	82.00	63.60	71.64 (+6.80)	82.74	65.91	73.37 (+3.98)
semi:word+top2	82.01	<i>67.52</i>	<i>74.06 (+9.22)</i>	<i>83.29</i>	<i>68.66</i>	<i>75.27 (+5.88)</i>

Figure 15: Named entity chunking results on the CoNLL’03 corpus. Without any gazetteer. For co-training and self-training (baseline), the best performance among all their parameter settings (including the number of iterations) is shown. (a) English, 10K labeled examples. (b) English, 204K (entire) labeled examples. (c) German, 207K (entire) labeled examples. The best performance in each column is italicized.

6. Discussions

This paper presents a general framework for learning predictive functional structures from multiple tasks. The idea is based on the concept that if multiple problems share a common predictive structure, then the structure can be more reliably estimated by considering these problems together. The process of learning the shared functional structure is referred to as structural learning. In the learning theory framework, structural learning is to discover a common structure of the hypothesis spaces shared by the problems. The main theoretical justification of this approach is that the shared structural parameter can be reliably estimated when m is large. Using the optimally estimated structural parameter, better generalization performance (averaged over the problems) can be achieved.

Moreover, we showed that the framework of structural learning can be applied to semi-supervised learning. This is achieved by creating auxiliary problems from unlabeled data that can reveal important underlying predictive structures of the data. Some examples of auxiliary problems were provided, and experimental results demonstrated that the discovered structures are very useful. Rigorously speaking, the theory we developed in Appendix A does not directly apply to semi-supervised learning. This is because in our theory, the performance is measured by averaged generalization ability over multiple prediction problems. However, in the setting of semi-supervised learning, we are only interested in the performance of the original supervised task, and not any of the auxiliary problems. For semi-supervised learning, a more relevant consequence of our analysis is that the shared structure can be stably estimated from multiple tasks. The usefulness of the shared structure is a different issue which is not directly answered by Appendix A. An intuitive justification of auxiliary problems we created is given in Section 4.3, although a more complete theory requires further investigation.

In summary, our approach to semi-supervised learning makes a bet on the existence of a shared predictive structure useful both for the supervised problem and for the auxiliary problems. The method proposed in Section 3 is robust since even if the discovered structure does not help on the supervised problem, the only potential negative effect is the introduction of some non-predictive features. Using typical discriminative learning methods with appropriate regularization, a small number of bad features only have a minor impact on the performance. However, if some of the features discovered from the auxiliary problems are useful, then the performance improvement can be significant.

The method derived in Section 3 has the intuitive interpretation of discovering low dimensional predictive structures on the classifier space. In our model, the most predictive dimensions correspond to the principal components of the multiple classifiers. Although our algorithm is based on the joint empirical risk minimization method which has a strong foundation in learning theory (see Appendix A), in principle, we can consider a more general approach of mining structures in the classifier space. Based on this general principle, one can design other structural learning algorithms that are not necessarily based on the joint empirical risk minimization method proposed in the paper. In fact, this general principle, which we may call *structural mining*, is the heart of our analysis. We shall thus conclude this paper by comparing some underlying concepts of structural mining to those of data-mining in Figure 16. In the table, a predictor can be regarded as a real-valued function defined on the data-space. The final row points out that we may also consider a data point

\mathbf{x} as a predictor on the predictor space by associating with each predictor p the functional value $\mathbf{x}(p) := p(\mathbf{x})$. In this sense, data-mining can be viewed as a special structural mining.

	data-mining	structural-mining
space of interest	data space	predictor space
instances	data-points	predictors from multiple tasks
uncertainty	measurement error	estimation error
goal	find patterns in data	find structures of the predictors
predictive power	maybe	yes
duality	a data point is a predictor of points in the predictor-space	

Figure 16: Data mining versus structural mining

Acknowledgments

The authors would like to thank Trevor Hastie and Robert Tibshirani for helpful discussions and for pointing out related statistical studies. Part of the work was supported by ARDA under the NIMD program PNWD-SW-6059.

Appendix A. Analysis of Structural Learning

We include a theoretical analysis of the joint empirical risk minimization method (3) for structure learning. The main purpose is to demonstrate that by joint minimization, the shared structure θ can be more reliably estimated. We consider the idealized case, where the performance of interests is the averaged loss over the m tasks. In particular, we are interested in the behavior when m becomes large. Our bound shows that using the joint empirical risk minimization method, it is possible to estimate the shared hypothesis space $\mathcal{H}_{\cdot,\theta}$ more reliably as $m \rightarrow \infty$.

Note that in practice, we are often interested in the performance on one particular task instead of the averaged performance over multiple tasks. This non-idealized scenario is not directly covered by our analysis. In particular, in the semi-supervised learning setting, additional theoretical analysis is needed to show that structure shared by the artificially created tasks can improve the performance of the supervised task (see Section 4.3). Still, the analysis presented here is relevant because it implies that the shared structure can be reliably estimated by the joint empirical risk minimization method, which we employ.

Instead of providing the most general analysis with the tightest possible generalization bounds, we adopt a relatively simple approach. Our purpose is to illustrate the main benefit of structural learning, that is, the ability to obtain an accurate estimate of the best hypothesis class $\mathcal{H}_{\cdot,\theta}$ ($\theta \in \Gamma$) when the number of problems m is large. The analysis is closely related to that of Baxter (2000) (also see Ben-David and Schuller, 2003). We use a different (although related) technical approach with a different covering number definition. The modifications are necessary to make our results directly applicable to the specific method proposed in Section 3.

For clarity, in the following analysis, we simplify (3) as

$$[\hat{\theta}, \{\hat{f}_\ell\}] = \arg \min_{\theta \in \Gamma, \{f_\ell \in \mathcal{H}_\theta\}} \sum_{\ell=1}^m \sum_{i=1}^n L(f_\ell(\mathbf{X}_i^\ell), Y_i^\ell), \quad (9)$$

where we only consider the case $n_1 = n_2 = \dots = n_m = n$, and $\mathcal{H}_{1,\theta} = \mathcal{H}_{2,\theta} = \dots = \mathcal{H}_{m,\theta} = \mathcal{H}_\theta$. This simplification is not critical, but allows us to consider the behavior of (9) as $m \rightarrow \infty$ under fixed n .

For simplicity, we use a covering-number approach in our analysis. The treatment is very similar to the case of $m = 1$, which is the standard empirical risk minimization. We need to introduce some definitions in order to state the main theorem.

Definition 1 Consider a set V with a distance function $d : V \times V \rightarrow \{0\} \cup \mathbb{R}^+$. Given $\epsilon > 0$, the ϵ -covering number of V , denoted by $\mathcal{N}(\epsilon, V, d(\cdot, \cdot))$, is the minimal number of balls $B(f) = \{g : d(f, g) \leq \epsilon\}$ of radius ϵ needed to cover V .

Definition 2 Let $S^{(n)} = \{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\}$ be a set of n points. We define the $\ell_2(S^{(n)})$ distance between any two functions $f(\mathbf{x}, y)$ and $g(\mathbf{x}, y)$ on $S^{(n)}$ as

$$\ell_2(S^{(n)})(f, g) = \left(\frac{1}{n} \sum_{i=1}^n |f(\mathbf{X}_i, Y_i) - g(\mathbf{X}_i, Y_i)|^2 \right)^{1/2}.$$

Let \mathcal{F} be a class of functions of (\mathbf{x}, y) . The empirical ℓ_2 -covering number of \mathcal{F} is the covering number $\mathcal{N}(\epsilon, \mathcal{F}, \ell_2(S^{(n)}))$ of \mathcal{F} with respect to the $\ell_2(S^{(n)})$ distance. The uniform ℓ_2 covering number is given by

$$\mathcal{N}_2(\epsilon, \mathcal{F}, n) = \sup_{S^{(n)}} \mathcal{N}(\epsilon, \mathcal{F}, \ell_2(S^{(n)})),$$

where the supremum is over all samples $S^{(n)}$ of size n .

Definition 3 Define distance d_∞ between hypothesis spaces H_θ ($\theta \in \Gamma$) as

$$d_\infty(\theta_1, \theta_2) = \sup_{f \in \mathcal{H}_{\theta_2}} \inf_{g \in \mathcal{H}_{\theta_1}} \sup_{\mathbf{x}, y} |f(\mathbf{x}, y) - g(\mathbf{x}, y)|.$$

We define the d_∞ -covering number of Γ as $\mathcal{N}(\epsilon, \Gamma, d_\infty)$.

The following theorem gives a (one-sided) uniform convergence result for the joint ERM method (9).

Theorem 4 For each $\ell = 1, \dots, m$, let $S_\ell = \{(\mathbf{X}_i^\ell, Y_i^\ell), \dots, (\mathbf{X}_n^\ell, Y_n^\ell)\}$ be a set of n points for problem ℓ , independently drawn from a distribution D_ℓ . Assume that $L(f(\mathbf{x}), y)$ is a bounded Lipschitz function of $f(\mathbf{x}) \in \mathcal{H}_\theta$. That is, there are θ -independent constants γ and M such that $\forall \theta_1, \theta_2 \in \Gamma$ and $\forall f_1 \in \mathcal{H}_{\theta_1}, f_2 \in \mathcal{H}_{\theta_2}$:

$$\begin{aligned} \forall(\mathbf{x}, y) : |L(f_1(\mathbf{x}), y) - L(f_2(\mathbf{x}), y)| &\leq \gamma |f_1(\mathbf{x}) - f_2(\mathbf{x})|, \\ \forall(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) : |L(f_1(\mathbf{x}_1), y_1) - L(f_1(\mathbf{x}_2), y_2)| &\leq M. \end{aligned}$$

Then there is a universal constant C such that $\forall \eta \in [0, 1]$, with probability $1 - \eta$, we have $\forall \hat{\theta}, \{\hat{f}_\ell \in H_{\hat{\theta}}\}$:

$$\frac{1}{m} \sum_{\ell=1}^m R_\ell(\hat{f}_\ell) \leq \frac{1}{m} \sum_{\ell=1}^m \hat{R}_\ell(\hat{f}_\ell, S_\ell) + \gamma C \inf_{\epsilon_0 \geq 0} \left[\epsilon_0 + \int_{\epsilon_0}^{\infty} \sqrt{\frac{\ln \mathcal{N}(\epsilon)}{n}} d\epsilon \right] + M \sqrt{\frac{\ln \frac{1}{\eta}}{nm}},$$

where R_ℓ and \hat{R}_ℓ are the true and empirical risks for problem ℓ :

$$R_\ell(\hat{f}_\ell) = \mathbf{E}_{(\mathbf{x}^\ell, Y^\ell) \sim D_\ell} L(\hat{f}_\ell(\mathbf{X}^\ell), Y^\ell), \quad \hat{R}_\ell(\hat{f}_\ell, S_\ell) = \frac{1}{n} \sum_{i=1}^n L(\hat{f}_\ell(\mathbf{X}_i^\ell), Y_i^\ell),$$

and

$$\ln \mathcal{N}(\epsilon) = \sup_{\theta} \ln \mathcal{N}_2(\epsilon, \mathcal{H}_\theta, n) + \frac{1}{m} \ln \mathcal{N}(\epsilon, \Gamma, d_\infty).$$

We shall delay the proof to the end of this appendix, and discuss the implications of the theorem first. In summary, this result justifies the joint ERM method (9), which minimizes the empirical risk on the right hand side of Theorem 4. The theorem implies that this method implicitly minimizes an upper bound of the true risk (averaged over the m problems) on the left hand side, which leads to a theoretical guarantee of the performance of this method.

The statistical complexity of the joint ERM method depends on the joint entropy $\ln \mathcal{N}(\epsilon)$, which has two components: the first term $\sup_{\theta} \ln \mathcal{N}_2(\epsilon, \mathcal{H}_\theta, n)$ is the learning complexity associated with individual estimation problems (with fixed θ). The second term $\frac{1}{m} \ln \mathcal{N}(\epsilon, \Gamma, d_\infty)$ is the complexity of estimating the best structural parameter θ . The most important consequence of our analysis is that the complexity of the structural space Γ , measured by the discounted entropy $\frac{1}{m} \ln \mathcal{N}(\epsilon, \Gamma, d_\infty)$, approaches zero when $m \rightarrow \infty$. This implies that we are able to find a near optimal (as measured by the generalization bound) shared structural parameter θ when m is large.

This theorem can be used to analyze the method we propose in Section 3, where in (4) and (5), a bi-linear structural model of the following form is used:

$$\mathcal{H}_\Theta = \left\{ \mathbf{w}^T \Phi(\mathbf{x}) + \mathbf{v}^T \Theta \Psi(\mathbf{x}) : \|\mathbf{w}\|_2 \leq \frac{A}{\sup_{\mathbf{x}} \|\Phi(\mathbf{x})\|_2}, \|\mathbf{v}\|_2 \leq \frac{B}{\sup_{\mathbf{x}} \|\Psi(\mathbf{x})\|_2} \right\},$$

$$\Gamma = \{\Theta \in R^{h \times p} : \Theta \Theta^T = I_{h \times h}\},$$

where $\Phi(\mathbf{x})$ and $\Psi(\mathbf{x})$ are pre-defined vector functions (feature maps) of \mathbf{x} ; \mathbf{v} is an h -dimensional vector; $\Psi(\mathbf{x})$ is a p -dimensional vector; and Θ is an orthonormal $h \times p$ dimensional matrix. We also use $I_{h \times h}$ to denote the h -dimensional identity matrix.

For this model, the matrix Θ , shared by the different prediction problems, is the structural parameter. When we fix Θ , the hypothesis space \mathcal{H}_Θ is parameterized by weight vectors \mathbf{w} and \mathbf{v} , where \mathbf{w} can be a high dimensional vector (regularized using A), and \mathbf{v} is a low dimensional vector (of dimensionality h). The idea of this model is to find a common low-dimensional predictive structure (shared by the m problems) parameterized by the projection matrix Θ . If we can discover such a structure, then we only need to use

a very small A to regularize the \mathbf{w} vector, which leads to improved generalization performance. In other words, there is a trade-off between the dimensionality h of the common low-dimensional predictive structure, and the regularization size A . The optimal trade-off is through the shared structural parameter Θ , which can be more reliably estimated using structural learning formulation (3) when m is large.

This intuitive argument can be more rigorously justified using Theorem 4 with appropriate covering number estimates. Specifically, it can be shown that there are universal constants C_1, C_2 and C_3 such that

$$\sup_{\theta} \ln \mathcal{N}_2(\epsilon, \mathcal{H}_{\theta}, n) \leq \frac{C_1 A^2}{\epsilon^2} + C_2 h \ln \left(1 + \frac{B}{\epsilon} \right), \quad \ln \mathcal{N}(\epsilon, \Gamma, d_{\infty}) \leq C_3 h p \ln \left(1 + \frac{B}{\epsilon} \right).$$

We shall not include a detailed proof of these estimates (which is not essential for the purpose of this paper) but only outline the basic ideas used in our derivation. The term $C_1 A^2 / \epsilon^2$ follows from a simple estimate of the Rademacher complexity of the sub function class in \mathcal{H}_{Θ} corresponding to $\mathbf{w}^T \Phi(\mathbf{x})$ as A / \sqrt{n} , and a straight-forward application of Sudak's minoration (e.g., see Ledoux and Talagrand, 1991, Chapter 12). The two $\ln(1 + B/\epsilon)$ terms can be obtained by explicit discretization of the corresponding finite dimensional parameter spaces — one for the h -dimensional sub function class in \mathcal{H}_{Θ} corresponding to $\mathbf{v}^T \Theta \Psi(\mathbf{x})$ (with fixed Θ and variable \mathbf{v}), and the other for a direct discretization of the hp -dimensional variable Θ . We simply note that a bounded set in a d -dimensional parameter space can be covered by $O(\epsilon^{-d})$ grid points with width no greater than ϵ in every direction.

By using the above covering number estimates, the complexity term in Theorem 4 becomes

$$\ln \mathcal{N}(\epsilon) \leq \frac{C_1 A^2}{\epsilon^2} + C_2 h \ln \left(1 + \frac{B}{\epsilon} \right) + \frac{1}{m} C_3 h p \ln \left(1 + \frac{B}{\epsilon} \right).$$

The third term is the complexity of estimating the structural parameter Θ , which vanishes as $m \rightarrow \infty$. The first and second terms characterize the trade-off between the regularization size A for the \mathbf{w} parameter, and the dimensionality h for the \mathbf{v} parameter. With the estimated Θ , the model approximates the underlying true predictor better for a fixed regularization size A (and thus a fixed complexity term $\ln \mathcal{N}(\epsilon)$ in Theorem 4), which implies better generalization behavior.

Proof of Theorem 4 Given training data $S = \cup_{\ell} S_{\ell}$, we define a vector function class on S :

$$\mathcal{F}_S = \{f = [f_{\ell}] : f(\mathbf{X}_i^{\ell}) = f_{\ell}(\mathbf{X}_i^{\ell}), f_{\ell} \in H_{\theta}, \theta \in \Gamma\},$$

where we use the notation $[f_{\ell}] = [f_{\ell}]_{\ell=1, \dots, m} = [f_1, \dots, f_m]$. Similarly, define

$$\mathcal{F}_S^L = \{[L(f(X_i^{\ell}), Y_i^{\ell})]_{\ell=1, \dots, m} : f \in \mathcal{F}_S\}.$$

We introduce two lemmas.

Lemma 5 *We have the following bounds:*

$$\ln \mathcal{N}(2\gamma\epsilon, \mathcal{F}_S^L, \ell_2(S)) \leq \ln \mathcal{N}(2\epsilon, \mathcal{F}_S, \ell_2(S)) \leq \sum_{\ell=1}^m \sup_{\theta} \ln \mathcal{N}(\epsilon, \mathcal{H}_{\theta}, \ell_2(S_{\ell})) + \ln \mathcal{N}(\epsilon, \Gamma, d_{\infty}).$$

Proof The first inequality is a direct consequence of the Lipschitz condition in Theorem 4. We shall prove the second inequality. Consider an ϵ -cover of Γ in the d_∞ metric. For simplicity, we denote the cover by $\theta_1, \dots, \theta_{N_\Gamma}$, where $N_\Gamma = \mathcal{N}(\epsilon, \Gamma, d_\infty)$. For each θ_j , we can find an ϵ -cover of H_{θ_j} on S_ℓ as $\bar{f}_{\ell,j,1}, \dots, \bar{f}_{\ell,j,N_\ell}$, where $N_\ell = \mathcal{N}(\epsilon, \mathcal{H}_\theta, \ell_2(S_\ell))$.

Now given any $f = [f_\ell] \in \mathcal{F}_S$, where $f_\ell \in \mathcal{H}_\theta$ for some $\theta \in \Gamma$, we can find $1 \leq J \leq N_\Gamma$, and $f' = [f'_\ell] \in \mathcal{F}_S$ such that each $f'_\ell \in \mathcal{H}_{\theta_J}$ and $\ell_2(S)(f, f') \leq \epsilon$. We can further approximate each f'_ℓ by a \bar{f}_{ℓ,J,K_ℓ} where $1 \leq K_\ell \leq N_\ell$ such that $\ell_2(S_\ell)(f'_\ell, \bar{f}_{\ell,J,K_\ell}) \leq \epsilon$. It follows that if we let $\bar{f} = [\bar{f}_{\ell,J,K_\ell}]_{\ell=1, \dots, m}$, then $\ell_2(S)(f', \bar{f}) \leq \epsilon$. Therefore we have $\ell_2(S)(f, \bar{f}) \leq 2\epsilon$. This means that \mathcal{F}_S has a 2ϵ -cover of the form $\bar{f} = [\bar{f}_{\ell,J,K_\ell}]$ ($J = 1, \dots, N_\Gamma$, $K_\ell = 1, \dots, N_\ell$ for $\ell = 1, \dots, m$). The size of this cover is $N_\Gamma \prod_\ell N_\ell$. \blacksquare

Lemma 6 *Let*

$$Q(S) = \sup_{[f_\ell] \in \mathcal{F}_S} \frac{1}{m} \sum_{\ell=1}^m (R_\ell(f_\ell) - \hat{R}_\ell(f_\ell, S_\ell)),$$

then $\forall \eta \in [0, 1]$, with probability $1 - \eta$:

$$Q(S) \leq \mathbf{E}_S Q(S) + M \sqrt{\frac{\ln \frac{1}{\eta}}{mn}}.$$

Proof For a given $1 \leq \bar{\ell} \leq m$ and $1 \leq \bar{i}$, we create a new dataset $\bar{S} = \cup_\ell \bar{S}_\ell$ by changing the \bar{i} -th datum of the $\bar{\ell}$ -th problem in $S = \cup_\ell S_\ell$ from $(X_{\bar{i}}^{\bar{\ell}}, Y_{\bar{i}}^{\bar{\ell}})$ to $(\bar{X}_{\bar{i}}^{\bar{\ell}}, \bar{Y}_{\bar{i}}^{\bar{\ell}})$ (and keep all the other data points identical). Then it is easy to verify that

$$Q(S) - Q(\bar{S}) \leq \sup_{\theta} \sup_{f \in \mathcal{H}_\theta} \frac{1}{mn} |L(f(X_{\bar{i}}^{\bar{\ell}}, Y_{\bar{i}}^{\bar{\ell}})) - L(f(\bar{X}_{\bar{i}}^{\bar{\ell}}, \bar{Y}_{\bar{i}}^{\bar{\ell}}))| \leq \frac{M}{mn}.$$

The lemma is a direct consequence of McDiarmid's concentration inequality (McDiarmid, 1989). \blacksquare

We are now ready to prove the main theorem. Consider a sequence of binary random variables $\sigma = \{\sigma_i^\ell\}$ such that each $\sigma_i^\ell = \pm 1$ is independent with probability 1/2. The Rademacher complexity of \mathcal{F}_S^L under empirical sample S , is given by

$$R(\mathcal{F}_S^L, S) = \mathbf{E}_\sigma \sup_{f \in \mathcal{F}_S} \left(\frac{1}{mn} \sum_{\ell=1}^m \sum_{i=1}^n \sigma_i^\ell L(f(\mathbf{X}_i^\ell), Y_i^\ell) \right).$$

It is well known that there exists a universal constant C (a variant of Corollary 2.2.8 in van der Vaart and Wellner, 1996):

$$R(\mathcal{F}_S^L, S) \leq \frac{C}{2} \inf_{\epsilon_0} \left[\epsilon_0 + \frac{1}{\sqrt{mn}} \int_{\epsilon_0}^{\infty} \sqrt{\ln \mathcal{N}_2(2\epsilon, \mathcal{F}_S^L, nm)} d\epsilon \right].$$

Applying Lemma 5, we obtain

$$R(\mathcal{F}_S^L, S) \leq \frac{C}{2} \inf_{\epsilon_0} \left[\epsilon_0 + \int_{\epsilon_0}^{\infty} \sqrt{\frac{\ln \mathcal{N}(\epsilon/\gamma)}{n}} d\epsilon \right] = \frac{\gamma C}{2} \inf_{\epsilon_0} \left[\epsilon_0 + \int_{\epsilon_0}^{\infty} \sqrt{\frac{\ln \mathcal{N}(\epsilon)}{n}} d\epsilon \right].$$

Using the standard symmetrization argument (for example, see Lemma 2.3.1 of (van der Vaart and Wellner, 1996)), we have

$$\mathbf{E}_S Q(S) \leq 2\mathbf{E}_S R(\mathcal{F}_S^L, S) = \gamma C \inf_{\epsilon_0} \left[\epsilon_0 + \int_{\epsilon_0}^{\infty} \sqrt{\frac{\ln \mathcal{N}(\epsilon)}{n}} d\epsilon \right].$$

The theorem is now a direct consequence of Lemma 6.

References

- R. K. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. In *ACL 05*, 2005.
- J. Baxter. A model for inductive bias learning. *Journal of Artificial Intelligent Research*, pages 149–198, 2000.
- M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, Special Issue on Clustering:209–239, 2004.
- S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *COLT 03*, 2003.
- A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.
- L. Breiman and J. Friedman. Predicting multivariate responses in multiple linear regression. *J. Roy. Statist. Soc. B.*, 59:3–37, 1997. with discussion.
- R. Caruana. Multi-task learning. *Machine Learning*, pages 41–75, 1997.
- H. L. Chieu and H. T. Ng. Named entity recognition with a maximum entropy approach. In *Proceedings CoNLL-2003*, pages 160–163, 2003.
- T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proc. Conf. on Knowledge Discovery and Data Mining*, 2004.
- R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *Proceedings CoNLL-2003*, pages 168–171, 2003.
- T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
- T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, 1999.

- D. Klein, J. Smarr, H. Nguyen, and C. D. Manning. Named entity recognition with character-level models. In *Proceedings CoNLL-2003*, pages 188–191, 2003.
- M. Ledoux and M. Talagrand. *Probability in Banach spaces*. Springer-Verlag, Berlin, 1991. ISBN 3-540-52013-9. Isoperimetry and processes.
- C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, pages 148–188. Cambridge University Press, 1989.
- C. A. Micchelli and M. Ponti. Kernels for multi-task learning. In *NIPS 2004*, 2005. to appear.
- K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, Special issue on information retrieval:103–134, 2000.
- D. Pierce and C. Cardie. Limitations of co-training for natural language learning from large datasets. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-2001)*, 2001.
- M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *NIPS 2001*, 2002.
- A. W. van der Vaart and J. A. Wellner. *Weak convergence and empirical processes*. Springer Series in Statistics. Springer-Verlag, New York, 1996. ISBN 0-387-94640-3.
- V. N. Vapnik. *Statistical learning theory*. John Wiley & Sons, New York, 1998.
- D. Yarowsky. unsupervised word sense disambiguation rivaling supervised methods. In *proceedings of ACL 95*, pages 189–196, 1995.
- T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 04*, pages 919–926, 2004.
- T. Zhang and D. E. Johnson. A robust risk minimization based named entity recognition system. In *Proceedings CoNLL-2003*, pages 204–207, 2003.
- T. Zhang and F. J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *ICML 00*, pages 1191–1198, 2000.
- D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS 2003*, pages 321–328, 2004.
- X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003*, 2003.

Feature Selection for Unsupervised and Supervised Inference: The Emergence of Sparsity in a Weight-Based Approach *

Lior Wolf †

*Center for Biological and Computational Learning
The McGovern Institute for Brain Research
Massachusetts Institute of Technology
Cambridge, MA, 02139 USA*

LIORWOLF@MIT.EDU

Amnon Shashua

*School of Computer Science and Engineering
The Hebrew University
Jerusalem 91904 Israel*

SHASHUA@CS.HUJI.AC.IL

Editor: Donald Geman

Abstract

The problem of selecting a subset of relevant features in a potentially overwhelming quantity of data is classic and found in many branches of science. Examples in computer vision, text processing and more recently bio-informatics are abundant. In text classification tasks, for example, it is not uncommon to have 10^4 to 10^7 features of the size of the vocabulary containing word frequency counts, with the expectation that only a small fraction of them are relevant. Typical examples include the automatic sorting of URLs into a web directory and the detection of spam email.

In this work we present a definition of “relevancy” based on spectral properties of the Laplacian of the features’ measurement matrix. The feature selection process is then based on a continuous ranking of the features defined by a least-squares optimization process. A remarkable property of the feature relevance function is that sparse solutions for the ranking values naturally emerge as a result of a “biased non-negativity” of a key matrix in the process. As a result, a simple least-squares optimization process converges onto a sparse solution, i.e., a selection of a subset of features which form a local maximum over the relevance function. The feature selection algorithm can be embedded in both unsupervised and supervised inference problems and empirical evidence show that the feature selections typically achieve high accuracy even when only a small fraction of the features are relevant.

1. Introduction

As visual recognition, text classification, speech recognition and more recently bio-informatics aim to address larger and more complex tasks the problem of focusing on the most relevant information in a potentially overwhelming quantity of data has become increasingly important. Examples from computer vision, text processing and Genomics are abundant. For instance, in visual recognition the pixel values themselves often form a highly redundant set of features; methods using an “over-complete” basis of features for recognition are gaining popularity (Olshausen and Field, 1996), and recently methods relying on abundance of simple efficiently computable features of which only

*. A short version of this paper was presented at the ICCV, Nice France, Oct. 2003.

†. The main body of this work was done while LW was at the Hebrew University.

a fraction of are relevant were proposed for face detection (Viola and Jones, 2001) — and these are only few examples from the visual recognition literature. In text classification tasks it is not uncommon to have 10^4 to 10^7 features of the size of the vocabulary containing word frequency counts, with the expectation that only a small fraction of them are relevant (Lewis, 1992). Typical examples include the automatic sorting of URLs into a web directory and the detection of spam email. In Genomics, a typical example is gene selection from micro-array data where the features are gene expression coefficients corresponding to the abundance of cellular mRNA taken from sample tissues. Typical applications include separating tumor from normal cells or discovery of new subclasses of Cancer cells based on the gene expression profile. Typically the number of samples (expression patterns) is less than 100 and the number of features (genes) in the raw data ranges from 5000 to 50000. Among the overwhelming number of genes only a small fraction is relevant for the classification of tissues whereas the expression level of many other genes may be irrelevant to the distinction between tissue classes — therefore, identifying highly relevant genes from the data is a basic problem in the analysis of expression data.

From a practical perspective, large amounts of irrelevant features affects learning algorithms at three levels. First, most learning problems do not scale well with the growth of irrelevant features — in many cases the number of training examples grows exponentially with the number of irrelevant features (Langley and Iba, 1993). Second, is a substantial degradation of classification accuracy for a given training set size (Almuallim and Dietterich, 1991; Kira and Rendell, 1992). The accuracy drop affects also advanced learning algorithms that generally scale well with the dimension of the feature space such as the Support Vector Machines (SVM) as recently observed in (Weston et al., 2001). The third aspect has to do with the run time of the learning algorithm on test instances. In most learning problems the classification process is based on inner-products between the features of the test instance and stored features from the training set, thus when the number of features is overwhelmingly large the run-time of the learning algorithm becomes prohibitively large for real time applications, for example. Another practical consideration is the problem of determining how many relevant features to select. This is a difficult problem which is hardly ever addressed in the literature and consequently it is left to the user to choose manually the number of features. Finally, there is an issue of whether one is looking for the *minimal* set of (relevant) features, or simply a possibly redundant but relevant set of features.

The potential benefits of feature selection include, first and foremost, better accuracy of the inference engine and improved scalability (defying the curse of dimensionality). Secondary benefits include better data visualization and understanding, reduce measurement and storage requirements, and reduce training and inference time. Blum and Langley (1997) in a survey article distinguish between three types of methods: *Embedded*, *Filter* and *Wrapper* approaches. The filter methods apply a preprocess which is independent of the inference engine (a.k.a the predictor or the classification/inference engine) and select features by ranking them with correlation coefficients or make use of mutual information measures. The Embedded and Wrapper approaches construct and select feature subsets that are useful to build a good predictor. The issue being the notion of *relevancy*, i.e., what constitutes a good set of features. The modern approaches, therefore, focus on building feature selection algorithms in the context of a *specific* inference engine. For example, (Weston et al., 2001; Bradley and Mangasarian, 1998) use the Support Vector Machine (SVM) as a subroutine (wrapper) in the feature selection process with the purpose of optimizing the SVM accuracy on the resulting subset of features. These wrapper and embedded methods in general are typically computationally expensive and often criticized as being “brute force”. Further details on relevancy versus usefulness

of features and references to historical and modern literature on feature selection can be found in the survey papers (Blum and Langley, 1997; Kohavi and John, 1997; Guyon and Elisseeff, 2003).

In this paper the inference algorithm is not employed directly in the feature selection process but instead general properties are being gathered which indirectly indicate whether a feature subset would be appropriate or not. Specifically, we use clustering as the predictor and use spectral properties of the candidate feature subset to guide the search. This leads to a “direct” approach where the search is conducted on the basis of optimizing desired spectral properties rather than on the basis of explicit clustering and prediction cycles. The search is conducted by the solution of a least-squares optimization function using a weighting scheme for the ranking of features. *A remarkable property of the energy function is that the feature weights come out positive as a result of a “biased non-negativity” of a key matrix in the process and sharply decay at the border between relevant and non-relevant features.* These properties make the algorithm ideal for “feature weighting” applications and for feature selection as the boundary between relevant and non-relevant features is typically clearly expressed by the decaying property of the feature weights. The algorithm, called $Q - \alpha$, is iterative, very efficient and achieves remarkable performance on a variety of experiments we have conducted.

There are many benefits of our approach: First, we avoid the expensive computations associated with Embedded and Wrapper approaches, yet still make use of a predictor to guide the feature selection. Second, the framework can handle both unsupervised and supervised inference within the same framework and handle any number of classes. In other words, since the underlying inference is based on clustering class labels are not necessary, but on the other hand, when class labels are provided they can be used by the algorithm to provide better feature selections. Third, the algorithm is couched within a least-squares framework — and least-squares problems are the best understood and easiest to handle. Finally, the performance (accuracy) of the algorithm is very good on a large number of experiments we have conducted.

2. Algebraic Definition of Relevancy

A key issue in designing a feature selection algorithm in the context of an inference is defining the notion of relevancy. Definitions of relevancy proposed in the past (Blum and Langley, 1997; Kohavi and John, 1997) lead naturally to an explicit enumeration of feature subsets which we would like to avoid. Instead, we take an algebraic approach and measure the relevance of a subset of features against its influence on the cluster arrangement of the data points with the goal of introducing an energy function which receives its optimal value on the desired feature selection. We will consider two measures of relevancy based on spectral properties where the first is based on the Standard spectrum and the second on the Laplacian spectrum.

2.1 The Standard Spectrum

Consider a $n \times q$ data set M consisting of q samples (columns) over n -dimensional feature space R^n representing n features x_1, \dots, x_n over q samples. Let the row vectors of M be denoted by $\mathbf{m}_1^\top, \dots, \mathbf{m}_n^\top$ pre-processed such that each row is centered around zero and is of unit L_2 norm $\|\mathbf{m}_i\| = 1$. Let $\mathcal{S} = \{x_{i_1}, \dots, x_{i_j}\}$ be a subset of (relevant) features from the set of n features and let $\alpha_i \in \{0, 1\}$ be the indicator value associated with feature x_i , i.e., $\alpha_i = 1$ if $x_i \in \mathcal{S}$ and zero otherwise (see Fig. 1). Let A_s be the corresponding *affinity* matrix whose (i, j) entries are the inner-product between the i 'th and j 'th data points restricted to the selected coordinate features, i.e., $A_s = \sum_{i=1}^n \alpha_i \mathbf{m}_i \mathbf{m}_i^\top$ where

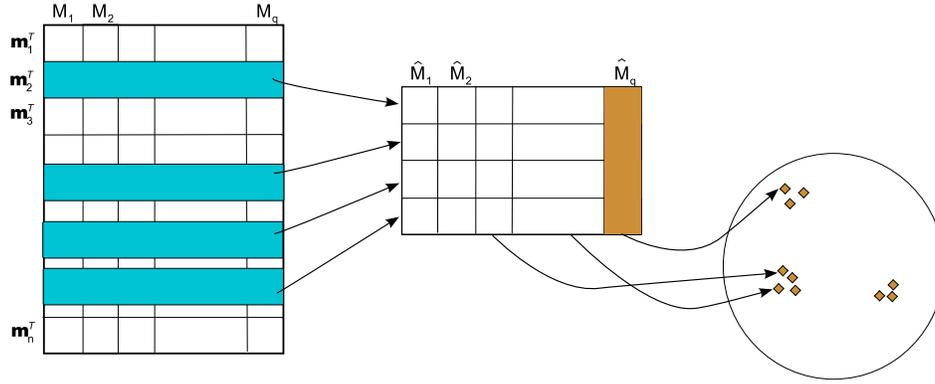


Figure 1: An illustration of variable-selection using our matrix notation. The large array on the left represents the matrix M , which contains q columns that represent the q data-points (M_1, \dots, M_q) . Each row of this matrix is a feature-vector $\mathbf{m}_1^\top, \dots, \mathbf{m}_n^\top$. In an idealized variable selection process, rows of the matrix M are selected to construct the matrix \hat{M} (middle), whose columns form well coherent clusters.

$\mathbf{m}_i \mathbf{m}_i^\top$ is the rank-1 matrix defined by the outer-product between \mathbf{m}_i and itself. Finally, let Q_s be a $q \times k$ matrix whose columns are the first k eigenvectors of A_s associated with the leading (highest) eigenvalues $\lambda_1 \geq \dots \geq \lambda_k$.

We define ‘relevancy’ as directly related to the clustering quality of the data points restricted to the selected coordinates. In other words, we would like to measure the quality of the subset S in terms of cluster coherence of the first k clusters, i.e., we make a direct linkage between cluster coherence of the projected data points and relevance of the selected coordinates.

We measure cluster coherence by analyzing the (standard) spectral properties of the affinity matrix A_s . Considering the affinity matrix as representing weights in an undirected graph, it is known that maximizing the quadratic form $\mathbf{x}^\top A_s \mathbf{x}$ where \mathbf{x} is constrained to lie on the standard simplex ($\sum x_i = 1$ and $x_i \geq 0$) provides the identification of the maximal *clique* of the (unweighted) graph (Motzkin and Straus, 1965; Gibbons et al., 1997), or the maximal ‘dominant’ subset of vertices of the weighted graph (Pavan and Pelillo, 2003). Likewise there is evidence (motivated by finding cuts in the graph) that solving the quadratic form above where \mathbf{x} is restricted to the unit sphere provides cluster membership information (cf. Ng et al., 2001; Weiss, 1999; Perona and Freeman, 1998; Shi and Malik, 2000; Brand and Huang, 2003; Chung, 1998). In this context, the eigenvalue (the value of the quadratic form) represents the cluster coherence. In the case of k clusters, the highest k eigenvalues of A_s represent the corresponding cluster coherences and the components of an eigenvector represent the coordinate (feature) participation in the corresponding cluster. The eigenvalues decrease as the interconnections of the points within clusters get sparser (see (Sarkar and Boyer, 1998)). Therefore, we define the relevance of the subset S as:

$$\begin{aligned} rel(S) &= \text{trace}(Q_s^\top A_s^\top A_s Q_s) \\ &= \sum_{r,s} \alpha_{i_r} \alpha_{i_s} (\mathbf{m}_{i_r}^\top \mathbf{m}_{i_s}) \mathbf{m}_{i_r}^\top Q_s Q_s^\top \mathbf{m}_{i_s} \end{aligned}$$

$$= \sum_{j=1}^k \lambda_j^2,$$

where λ_j are the leading eigenvalues of A_s . Note that the proposed measure of relevancy handles interactions among features up to a second order. To conclude, achieving a high score on the combined energy of the first k eigenvalues of A_s indicate (although indirectly) that the q input points projected onto the l -dimensional feature space are “well clustered” and that in turn suggests that S is a relevant subset of features.

Maximizing the relevancy above for all possible feature subsets is infeasible. Therefore, we relax the problem, i.e., instead of enumerating the feature subsets S and ranking them according to the value of $rel(S)$ we consider the prior weights $\alpha_1, \dots, \alpha_n$ as unknown *real numbers* and define the following optimization function:

Definition 1 (Relevant Features Optimization) *Let M be an $n \times q$ input matrix with rows $\mathbf{m}_1^\top, \dots, \mathbf{m}_n^\top$. Let $A_\alpha = \sum_{i=1}^n \alpha_i \mathbf{m}_i \mathbf{m}_i^\top$ for some unknown scalars $\alpha_1, \dots, \alpha_n$. The weight vector $\alpha = (\alpha_1, \dots, \alpha_n)^\top$ and the orthonormal $q \times k$ matrix Q are determined at the maximal point of the following optimization problem:*

$$\begin{aligned} \max_{Q, \alpha_i} & \text{trace}(Q^\top A_\alpha^\top A_\alpha Q) & (1) \\ \text{subject to} & \sum_{i=1}^n \alpha_i^2 = 1, \quad Q^\top Q = I \end{aligned}$$

Note that the optimization function does not include the inequality constraint $\alpha_i \geq 0$ and neither a term for “encouraging” a sparse solution of the weight vector α — both of which are necessary for a “feature selection”. As will be shown later in Section 4, the sparsity and positivity conditions are implicitly embedded in the nature of the optimization function and therefore “emerge” naturally with the optimal solution.

Note also that it is possible to maximize the gap $\sum_{i=1}^k \lambda_i^2 - \sum_{j=k+1}^q \lambda_j^2$ by defining $Q = [Q_1 | Q_2]$ where Q_1 contains the first k eigenvectors and Q_2 the remaining $q - k$ eigenvectors (sorted by decreasing eigenvalues) and the criterion function (1) would be replaced by:

$$\max_{Q=[Q_1|Q_2], \alpha_i} \text{trace}(Q_1^\top A_\alpha^\top A_\alpha Q_1) - \text{trace}(Q_2^\top A_\alpha^\top A_\alpha Q_2).$$

We will describe in Section 3 an efficient algorithm for finding a local maximum of the optimization (1) and later address the issue of sparsity and positivity of the resulting weight vector α . The algorithms are trivially modified to handle the gap maximization criterion and those will not be further elaborated here. We will describe next the problem formulation using an additive normalization (the Laplacian) of the affinity matrix.

2.2 The Laplacian Spectrum

Given the standard affinity matrix A , consider the Laplacian matrix: $L = A - D + d_{max}I$ where D is a diagonal matrix $D = \text{diag}(\sum_j a_{ij})$ and d_{max} is a scalar larger or equal to the maximal element of D .¹ The matrix L normalizes A in an additive manner and there is much evidence to support such

1. Note that in applications of algebraic graph theory the Laplacian is defined as $D - A$. The reason for the somewhat different definition is that we wish to maintain the order of eigenvectors as in those of A (where the eigenvectors associated with the largest eigenvalues come first).

a normalization both in the context of graph partitioning (Mohar, 1991; Hall, 1970) and spectral clustering (Weiss, 1999; Ng et al., 2001).

It is possible to reformulate the feature selection problem (1) using the Laplacian as follows. Let $A_i = \mathbf{m}_i \mathbf{m}_i^\top$ and $D_i = \text{diag}(\mathbf{m}_i \mathbf{m}_i^\top \mathbf{1})$. We define $L_\alpha = \sum_i \alpha_i L_i$ where $L_i = A_i - D_i + d_{\max} I$. We have, therefore:

$$L_\alpha = A_\alpha - D_\alpha + \left(\sum_i \alpha_i \right) d_{\max} I,$$

where $D_\alpha = \text{diag}(A_\alpha^\top \mathbf{1})$. Note that since α is a unit norm vector that contains positive elements, then $\sum_i \alpha_i > 1$. The feature selection problem is identical to (1) where L_α replaces A_α .

3. An Efficient Algorithm

We wish to find an optimal solution for the non-linear problem (1). We will focus on the Standard spectrum matrix A_α and later discuss the modifications required for L_α . If the weight vector α is known, then the solution for the matrix Q is readily available by employing a Singular Value Decomposition (SVD) of the symmetric (and positive definite) matrix A_α . Conversely, if Q is known then α is readily determined as shown next. We already saw that

$$\begin{aligned} \text{trace}(Q^\top A_\alpha^\top A_\alpha Q) &= \sum_{i,j} \alpha_i \alpha_j (\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top Q Q^\top \mathbf{m}_j \\ &= \alpha^\top G \alpha \end{aligned}$$

where $G_{ij} = (\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top Q Q^\top \mathbf{m}_j$ is symmetric and positive definite. The optimal α is therefore the solution of the optimization problem:

$$\max_{\alpha} \alpha^\top G \alpha \quad \text{subject to } \alpha^\top \alpha = 1,$$

which results in α being the leading eigenvector of G , i.e., the one associated with its largest eigenvalue. A possible scheme, guaranteed to converge to a local maxima, is to start with some initial guess for α and iteratively interleave the computation of Q given α and the computation of α given Q until convergence. We refer to this scheme as the **Basic $Q - \alpha$ Method**.

In practice, the number of iterations is rather small — typically between 5 to 10. The runtime complexity as a function of the number of features n is therefore governed by the complexity of finding the leading eigenvector of G — typically in the order of n^2 assuming a “reasonable” spectral gap (for example, if G were a random matrix then the spectral gap is large — asymptotically in the order of \sqrt{n} — as we know from the semi-circle law (Wigner, 1958)). A quadratic complexity is the best that one can expect when performing feature selection in an unsupervised manner since all pairs of feature vectors need to be compared to each other.

A more advanced scheme with superior convergence rate and more importantly accuracy of results (based on empirical evidence) is to embed the computation of α within the “orthogonal iteration” (Golub and Loan, 1996) cycle for computing the largest k eigenvectors, described below:

Definition 2 (Standard Power-Embedded $Q - \alpha$ Method) *Let M be an $n \times q$ input matrix with rows $\mathbf{m}_1^\top, \dots, \mathbf{m}_n^\top$, and some orthonormal $q \times k$ matrix $Q^{(0)}$, i.e., $Q^{(0)\top} Q^{(0)} = I$. Perform the following steps through a cycle of iterations with index $r = 1, 2, \dots$*

1. Let $G^{(r)}$ be a matrix whose (i, j) components are $(\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top Q^{(r-1)} Q^{(r-1)\top} \mathbf{m}_j$.

2. Let $\alpha^{(r)}$ be the largest eigenvector of $G^{(r)}$.
3. Let $A^{(r)} = \sum_{i=1}^n \alpha_i^{(r)} \mathbf{m}_i \mathbf{m}_i^\top$.
4. Let $Z^{(r)} = A^{(r)} Q^{(r-1)}$.
5. $Z^{(r)} \xrightarrow{QR} Q^{(r)} R^{(r)}$.
6. Increment index r and go to step 1.

The method is considerably more efficient than the basic scheme above and achieves very good performance (accuracy). Note that steps 4,5 of the algorithm consist of the ‘‘orthogonal iteration’’ module, i.e., if we were to repeat steps 4,5 *only* we would converge onto the eigenvectors of $A^{(r)}$. However, note that the algorithm does not repeat steps 4,5 in isolation and instead recomputes the weight vector α (steps 1,2,3) before applying another cycle of steps 4,5. We show below that the recomputation of α does not alter the convergence property of the orthogonal iteration scheme, thus the overall scheme converges to a local maxima:

Proposition 3 (Convergence of Power-Embedded $Q - \alpha$) *The Power Embedded $Q - \alpha$ method convergence to a local maxima of the criterion function (1).*

Proof: We will prove the claim for the case $k = 1$, i.e., the scheme optimizes over the weight vector α and the largest eigenvector \mathbf{q} of A_α .

Because the computation of α is analytic (the largest eigenvector of G) and because the optimization energy is bounded from above, it is sufficient to show that the computation of \mathbf{q} monotonically increases the criterion function. It is therefore sufficient to show that:

$$\mathbf{q}^{(r)} A^2 \mathbf{q}^{(r)} \geq \mathbf{q}^{(r-1)} A^2 \mathbf{q}^{(r-1)}, \quad (2)$$

for all symmetric matrices A . Since steps 4,5 of the algorithm are equivalent to the step:

$$\mathbf{q}^{(r)} = \frac{A \mathbf{q}^{(r-1)}}{\|A \mathbf{q}^{(r-1)}\|},$$

we can substitute the right hand side into (2) and obtain the condition:

$$\mathbf{q}^\top A^2 \mathbf{q} \leq \frac{\mathbf{q}^\top A^4 \mathbf{q}}{\mathbf{q}^\top A^2 \mathbf{q}}, \quad (3)$$

which needs to be shown to hold for all symmetric matrices A and unit vectors \mathbf{q} . Let $\mathbf{q} = \sum_i \gamma_i \mathbf{v}_i$ be represented with respect to the orthonormal set of eigenvectors \mathbf{v}_i of the matrix A . Then, $A \mathbf{q} = \sum_i \gamma_i \lambda_i \mathbf{v}_i$ where λ_i are the corresponding eigenvalues. Since $\mathbf{q}^\top A^2 \mathbf{q} \geq 0$, it is sufficient to show that: $\|A \mathbf{q}\|^4 \leq \|A^2 \mathbf{q}\|^2$, or equivalently:

$$\left(\sum_i \gamma_i^2 \lambda_i^2 \right)^2 \leq \sum_i \gamma_i^2 \lambda_i^4. \quad (4)$$

Let $\mu_i = \lambda_i^2$ and let $f(x) = x^2$. We then have:

$$f\left(\sum_i \gamma_i^2 \mu_i\right) \leq \sum_i \gamma_i^2 f(\mu_i),$$

which follows from convexity of $f(x)$ and the fact that $\sum_i \gamma_i^2 = 1$. \square

A faster converging algorithm is possible by employing the ‘‘Ritz’’ acceleration (Golub and Loan, 1996) to the basic power method as follows:

Definition 4 ($Q - \alpha$ with Ritz Acceleration) Let M be an $n \times q$ input matrix with rows $\mathbf{m}_1^\top, \dots, \mathbf{m}_n^\top$, and some orthonormal $n \times k$ matrix $Q^{(0)}$, i.e., $Q^{(0)\top} Q^{(0)} = I$. Perform the following steps through a cycle of iterations with index $r = 1, 2, \dots$

1. Let $G^{(r)}, \alpha^{(r)}$ and $A^{(r)}$ be defined as in the Standard Power-Embedded $Q - \alpha$ algorithm.
2. $Z^{(r)} = A^{(r)} Q^{(r-1)}$.
3. $Z^{(r)} \xrightarrow{QR} \bar{Q}^{(r)} R^{(r)}$.
4. Let $\bar{G}^{(r)}$ be a matrix whose (i, j) components are $\mathbf{m}_i^\top \bar{Q}^{(r)\top} \bar{Q}^{(r)} \mathbf{m}_j$.
5. Recompute $\alpha^{(r)}$ as the largest eigenvector of $\bar{G}^{(r)}$, and recompute $A^{(r)}$ accordingly.
6. Let $S^{(r)} = \bar{Q}^{(r)\top} A^{(r)} \bar{Q}^{(r)}$.
7. Perform SVD on $S^{(r)}$: $[U^{(r)\top} S^{(r)} U^{(r)}] = \text{svd}(S^{(r)})$.
8. $Q^{(r)} = \bar{Q}^{(r)} U^{(r)}$.
9. Increment index r and go to step 1.

The $Q - \alpha$ algorithm for the Laplacian spectrum L_α follows the Standard spectrum with the necessary modifications described below.

Definition 5 (Laplacian Power-Embedded $Q - \alpha$ Method) In addition to the definition of the Standard method, let $d_i = \max \text{diag}(\mathbf{m}_i \mathbf{m}_i^\top)$ and $L_i^{(0)} = \mathbf{m}_i \mathbf{m}_i^\top - \text{diag}(\mathbf{m}_i \mathbf{m}_i^\top \mathbf{1}) + d_i I$. Perform the following steps with index $r = 1, 2, \dots$

1. Let $F^{(r)}$ be a matrix whose (i, j) components are $\text{trace}(Q^{(r-1)\top} L_i^{(r-1)\top} L_j^{(r-1)} Q^{(r-1)})$.
2. Let $\alpha^{(r)}$ be the largest eigenvector of $F^{(r)}$.
3. Let $d^{(r)} = (\max \text{diag}(\sum_{i=1}^n \alpha_i^{(r)} \mathbf{m}_i \mathbf{m}_i^\top)) / (\sum_{i=1}^n \alpha_i)$
4. For each i let $L_i^{(r)} = \mathbf{m}_i \mathbf{m}_i^\top - \text{diag}(\mathbf{m}_i \mathbf{m}_i^\top \mathbf{1}) + d^{(r)} I$
5. Let $L^{(r)} = \sum_{i=1}^n \alpha_i^{(r)} L_i^{(r)}$.
6. Let $Z^{(r)} = L^{(r)} Q^{(r-1)}$.
7. $Z^{(r)} \xrightarrow{QR} Q^{(r)} R^{(r)}$.
8. Increment index r and go to step 1.

3.1 The Supervised Case

The $Q - \alpha$ algorithms and the general approach can be extended to handle data with class labels. One of the strengths of our approach is that the feature selection method can handle both unsupervised and supervised data sets. In a nutshell, the supervised case is handled as follows. Given c classes, we are given c data matrices $M^l, l = 1, \dots, c$, each of size $n \times q^l$.

Definition 6 (Supervised Relevant Features Optimization) Let M^l be an $n \times q^l$ input matrices with rows $\mathbf{m}_1^{l\top}, \dots, \mathbf{m}_n^{l\top}$. Let $A_\alpha^{gh} = \sum_{i=1}^n \alpha_i \mathbf{m}_i^g \mathbf{m}_i^{h\top}$ for some unknown scalars $\alpha_1, \dots, \alpha_n$. The weight vector $\alpha = (\alpha_1, \dots, \alpha_n)^\top$ and the orthonormal $q^h \times k^{gh}$ matrices Q^{gh} are determined at the maximal point of the following optimization problem:

$$\begin{aligned}
 & \max_{Q^{gh}, \alpha_i} \sum_l \text{trace}(Q^{ll\top} A_\alpha^{ll\top} A_\alpha^{ll} Q^{ll}) \\
 & - \gamma \sum_{g \neq h} \text{trace}(Q^{gh\top} A_\alpha^{gh\top} A_\alpha^{gh} Q^{gh}) \\
 & \text{subject to } \sum_{i=1}^n \alpha_i^2 = 1, \quad Q^{gh\top} Q^{gh} = I
 \end{aligned} \tag{5}$$

Where the weight γ and the parameters k^{gh} are determined manually (see below).

The criterion function seeks a weight vector α such that the resulting affinity matrix of all the data points (sorted) would be semi-block-diagonal, i.e., high inter-class eigenvalue energy and low intra-class energy. Therefore, we would like to minimize of the intra-class eigenvalue energy $\text{trace}(Q^{gh\top} A_\alpha^{gh\top} A_\alpha^{gh} Q^{gh})$ (off-block-diagonal blocks) and maximize the inter-class eigenvalue energy $\text{trace}(Q^{ll\top} A_\alpha^{ll\top} A_\alpha^{ll} Q^{ll})$. The parameters k^{gh} control the complexity of each affinity matrix. A typical choice of the parameters would be $k^{gh} = 2$ when $g = h$, $k^{gh} = 1$ otherwise, and $\gamma = 0.5$.

The solution to the optimization function follows step-by-step the $Q - \alpha$ algorithms. At each cycle Q^{gh} is computed using the current estimates A_α^{gh} and α is optimized by maximizing the expression:

$$\sum_l \alpha^\top G^{ll} \alpha - \gamma \sum_{g \neq h} \alpha^\top G^{gh} \alpha = \alpha^\top \mathcal{G} \alpha \quad ,$$

where $G_i^{gh} = (\mathbf{m}_i^{g\top} \mathbf{m}_j^g) \mathbf{m}_i^{h\top} Q^{gh\top} Q^{gh} \mathbf{m}_j^h$ and $\mathcal{G} = \sum_l G^{ll} - \gamma \sum_{g \neq h} G^{gh}$. We analyze next the properties of the unsupervised $Q - \alpha$ algorithm with regard to sparsity and positivity of the weight vector α and then proceed to experimental analysis.

4. Sparsity and Positivity of α

The optimization criteria (1) is formulated as a least-squares problem and as such there does not seem to be any apparent guarantee that the weights $\alpha_1, \dots, \alpha_n$ would come out *non-negative* (same sign condition), and in particular *sparse* when there exists a sparse solution (i.e., there is a relevant subset of features which induces a coherent clustering).

The positivity of the weights is a critical requirement for the $Q - \alpha$ to form a “feature weighting” scheme. In other words, if one could guarantee that the weights would come out non-negative then $Q - \alpha$ would provide feature weights which could be used for selection or for simply weighting the features as they are being fed into the inference engine of choice. If in addition the feature weights exhibit a “sparse” profile, i.e., the gap between the high and low values of the weights is high, then the weights could be used for selecting the relevant features as well. We will refer to the gap between the high and low weights as “sparsity gap” and discuss later in the paper the value of the gap in simplified domains. With the risk of abusing standard terminology, we will refer to the property of having the weight vector concentrate its (high) values around a number of coordinates as a sparsity feature. Typically, for our algorithm, none of the values of the weight vector strictly vanish.

For most feature weighting schemes, the conditions of positivity and sparsity should be specifically presented into the optimization criterion one way or the other. The possible means for doing so include introduction of inequality constraints, use of L_0 or L_1 norms, adding specific terms to the optimization function to “encourage” sparse solutions or use a multiplicative scheme of iterations which preserve the sign of the variables throughout the iterations (for a very partial list see

Olshausen and Field, 1996; Kivinen and Warmuth, 1997; Lee and Seung, 1999; Vapnik, 1998). It is therefore somewhat surprising, if not remarkable, that the least-squares formulation of the feature selection problem could consistently converge onto same-sign and sparse solutions.

Before we proceed with the technical issues, it is worthwhile to make qualitative arguments (which were the basis of developing this approach to begin with) as to the underlying reason for sparsity. Consider rewriting the optimization criterion (1) by an equivalent criterion:

$$\min_{\alpha, Q} \left\{ \|A_\alpha - QQ^\top A_\alpha\|_F^2 - \|A_\alpha\|_F^2 \right\} \quad (6)$$

where $\|\cdot\|_F^2$ is the square Frobenius norm of a matrix defined as the sum of squares of all entries of the matrix. The first term of (6) measures the distance between the columns of A_α and the projection of those columns onto a k -dimensional subspace (note that QQ^\top is a projection matrix). This term receives a low value if indeed A_α has a small (k) number of dominant eigenvectors, i.e., the spectral properties of the feature subset represented by A_α are indicative to a good clustering score. Since $A_\alpha = \sum_i \alpha_i \mathbf{m}_i \mathbf{m}_i^\top$ is represented by the sum of rank-1 matrices one can combine only a *small* number of them if the first term is desired to be small. The second term (which may be viewed also as a regularization term) encourages addition of more rank-1 matrices to the sum provided they are *redundant*, i.e., are already spanned by the previously selected rank-1 matrices. This makes the point that the feature selection scheme looks for relevant features but not necessarily the minimal set of relevant features. To summarize, from a qualitative point of view the selection of values for the weights α_i is directly related to the rank of the affinity matrix A_α which should be small if indeed A_α arises from a clustered configuration of data points. A uniform spread of values α_i would result in a high rank for A_α , thus the criteria function encourages a non-uniform (i.e., sparse) spread of weight values.

The argument presented above to facilitate clarity of the approach and should not be taken as a proof for sparsity. The positivity and sparsity issues are approached in the sequel from a different angle which provides a more analytic handle to the underlying search process than the qualitative argument above.

4.1 Positivity of α

The key for the emergence of a sparse and positive α has to do with the way the entries of the matrix G are defined. Recall that $G_{ij} = (\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top QQ^\top \mathbf{m}_j$ and that α comes out as the leading eigenvector of G (at each iteration). If G were to be non-negative (and irreducible), then from the Perron-Frobenius theorem the leading eigenvector is guaranteed to be non-negative (or same-sign). However, this is not the case and G in general has negative entries as well as positive ones. However, from a probabilistic point of view the probability that the leading eigenvector of G will come out positive rapidly approaches 1 with the growth of the number of features — this under a couple of simplifying assumptions.

The simplifying assumptions we will make in order to derive a probabilistic argument, is first that the entries of the upper triangular part of G are independent. The second simplifying approximation is that the columns of Q are sampled uniformly over the unit hypersphere. Although the independence and uniformity assumptions are indeed an idealization of the true nature of G and Q , they nevertheless allow us to derive a powerful probabilistic argument which shows in a rigorous manner that the weights α_i are non-negative with probability 1 — a statement which agrees with practice over extensive experimentations which we have performed.

The probabilistic approach follows from the observation that each entry of G consists of a sum of products of three inner-products:

$$G_{ij} = \sum_{l=1}^k (\mathbf{m}_i^\top \mathbf{q}_l)(\mathbf{m}_j^\top \mathbf{q}_l)(\mathbf{m}_i^\top \mathbf{m}_j).$$

In general, a product of the form $f = (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{c})$, where $\|\mathbf{a}\| = \|\mathbf{b}\| = \|\mathbf{c}\| = 1$ satisfies $-1/8 \leq f \leq 1$ where $f = 1$ when $\mathbf{a} = \mathbf{b} = \mathbf{c}$. Since $f \geq -1/8$ (will be proven below) there is an asymmetry on the expected value of f , i.e., the expected values of the entries of G are biased towards a positive value — and we should expect a bias towards a positive leading eigenvector of G . We will derive below the expectation on the entries of G (assuming independence and uniformity) and prove the main theorem showing that a random matrix whose entries are sampled i.i.d. from some distribution with positive mean and bounded variance has a positive leading eigenvector with probability 1 when the number of features n is sufficiently large. The details are below.

Proposition 7 *The minimal value of $f = (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{c})$ where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^q$ are defined over the unit hypersphere is $-1/8$.*

Proof: See appendix.

Proposition 8 *The expected value of $f = (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{c})$ where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^q$ and \mathbf{c} is uniformly sampled over the unit hypersphere is $(1/q)(\mathbf{a}^\top \mathbf{b})^2$*

Proof: See appendix.

To get a rough estimate on the values in the matrix G we can further assume that \mathbf{a} and \mathbf{b} are also evenly distributed on the q -dim sphere. In this case the expectation of $(\mathbf{a}^\top \mathbf{b})^2$ is $1/q$. To see this observe that the expectation $E(\mathbf{a}^\top \mathbf{b})^2 = \int \int (\mathbf{a}^\top \mathbf{b})^2 d\sigma(\mathbf{a}) d\sigma(\mathbf{b}) = \int \mathbf{a}^\top (\int \mathbf{b} \mathbf{b}^\top d\sigma(\mathbf{b})) \mathbf{a} d\sigma(\mathbf{a}) = \int \mathbf{a}^\top ((1/q)I) \mathbf{a} d\sigma(\mathbf{a})$ where I is the identity matrix in \mathbb{R}^q .

Each entry G_{ij} is a sum of k such terms, $G_{ij} = \sum_{l=1}^k (\mathbf{m}_i^\top \mathbf{q}_l)(\mathbf{m}_j^\top \mathbf{q}_l)(\mathbf{m}_i^\top \mathbf{m}_j)$. If the features are irrelevant, we can expect the correlation with the vector \mathbf{q}_1 to be similar to correlation with a “typical” random vector. In this case the above proposition applies. However, when $k > 1$ there are interrelations between the elements in the sum resulting from the orthogonality of the columns of Q . The following proposition shows that the expectation is still larger than zero.

Proposition 9 *The expected value of $f = \sum_{i=1}^k (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c}_i)(\mathbf{b}^\top \mathbf{c}_i)$ where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^q$ and \mathbf{c}_i are orthonormal vectors uniformly sampled over the unit hypersphere in \mathbb{R}^q is $(k/q)(\mathbf{a}^\top \mathbf{b})^2$.*

Proof: See appendix.

The body of results on spectral properties of random matrices (see for example Mehta, 1991) deals with the distribution of eigenvalues. For example, the corner-stone theorem known as Wigner’s *semicircle* theorem (Wigner, 1958) is about the asymptotic distribution of eigenvalues with the following result: “Given a symmetric $n \times n$ matrix whose entries are bounded independent random variables with mean μ and variance σ^2 , then for any $c > 2\sigma$, with probability $1 - o(1)$ all eigenvalues except for at most $o(n)$ belong to $\Theta(c\sqrt{n})$, i.e., lie in the interval $I = (-c\sqrt{n}, c\sqrt{n})$.”

The notation $f(n) = o(g(n))$ stands for $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$, i.e., $f(n)$ becomes insignificant relative to $g(n)$ with the growth of n . This is a short-hand notation (which we will use in the sequel) to the formal statement: “ $\forall \epsilon > 0, \exists n_0$ s.t. $\forall n > n_0$ the statement holds with probability $1 - \epsilon$.”

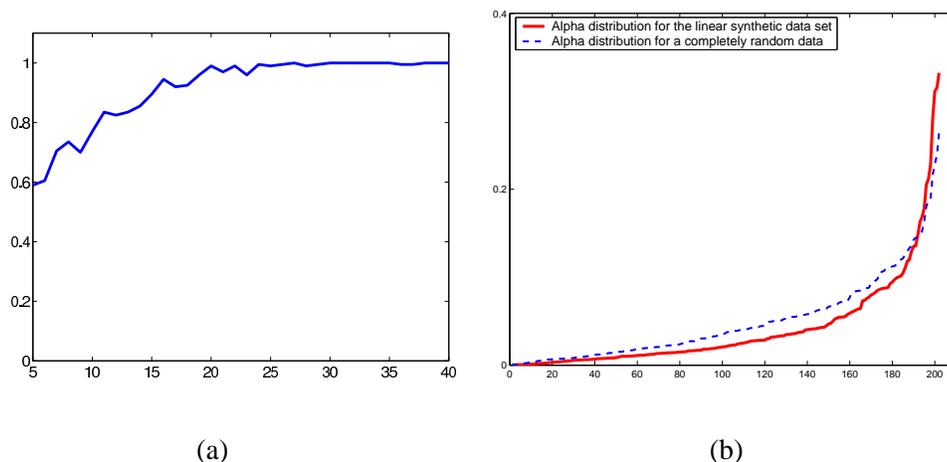


Figure 2: (a) Probability, as computed using simulations, of positive leading eigenvector of a symmetric matrix G with i.i.d elements drawn from the Gaussian distribution with $\mu = 1/6$ and $\sigma = \sqrt{2}/6$. The probability is very close to 1 starting from $n = 20$. (b) Positivity and sparsity demonstrated on the synthetic feature selection problem described in Section 6 (6 relevant features out of 202) and of a random data matrix. The alpha weight vector (sorted for display) comes out positive and sparse.

It is also known that when $\mu = 0$ all the eigenvalues belong to the interval I (with probability $1 - o(1)$), while for the case $\mu > 0$ only the leading eigenvalue λ_1 is outside of I and

$$\lambda_1 = \frac{1}{n} \sum_{i,j} G_{ij} + \frac{\sigma^2}{\mu} + O\left(\frac{1}{\sqrt{n}}\right),$$

i.e., λ_1 asymptotically has a normal distribution with mean $\mu n + \sigma^2/\mu$ (Furedi and Komlos, 1981). Our task is to derive the asymptotic behavior of the leading eigenvector when $\mu > 0$ under the assumption that the entries of G are i.i.d. random variables. We will first prove the theorem below, which deals with Gaussian random variables, and then extend it to bounded random variables:

Theorem 10 (Probabilistic Perron-Frobenius) *Let $G = g_{ij}$ be a real symmetric $n \times n$ matrix whose entries for $i \geq j$ are independent identically and normally distributed random variables with mean $\mu > 0$ and variance σ^2 . Then, for any $\epsilon > 0$ there exist n_0 such that for all $n > n_0$ the leading eigenvector \mathbf{v} of G is positive with probability of at least $1 - \epsilon$.*

Proof: see appendix.

Fig. 2(a) displays a simulation result plotting the probability of positive leading eigenvector of G (with $\mu = 1/6$ and $\sigma = \sqrt{2}/6$) as a function of n . One can see that for $n > 20$ the probability becomes very close to 1 (above 0.99). Simulations with $\mu = 0.1$ and $\sigma = 1$ show that the probability is above 0.99 starting from $n = 500$.

Theorem 10 used independent Gaussian random variables as a model to the matrix G . This might seem a bit artificial, since the variables of the matrix G are dependent and bounded. While

the independence assumption between all the elements in the upper triangular part of G is hard to remove, the use of Gaussian variables is not essential; as stated above the semi circle law holds for matrices with elements that are not necessarily Gaussian.

The only place where we actually used the ‘‘Gaussianity’’ property was in the assumed structure of the variable \mathbf{g} . Since \mathbf{g} contains normal i.i.d distributions, we deduced that $\|\mathbf{g}\| = \Theta(\sqrt{n})$ and that the probability of $\|\mathbf{g}\| \geq n^{3/4}$ decays exponentially. Instead of Gaussianity, we can use the property that the elements of the matrix G are bounded, and instead of Gaussian tail bounds we can use Hoeffding’s tail inequality (Hoeffding, 1963). We will use the one sided inequality ²

Lemma 11 (Hoeffding’s one-sided tail inequality) *Let X_1, X_2, \dots, X_n be bounded independent random variables such that $X_i \in [a_i, b_i]$. Then for $S_n = X_1 + X_2 + \dots + X_n$ the following inequality holds*

$$Pr(S_n - ES_n \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

Using Hoeffding’s lemma, the following lemma could be used to bound the norm of \mathbf{g} .

Lemma 12 *Let \mathbf{g} be a random n -vector of i.i.d bounded variables, i.e., for each $i = 1..n$, $|\mathbf{g}_i| \leq M$. The following holds for some constant C :*

$$P(\|\mathbf{g}\|^2 \geq Dn^{1/2+\epsilon}) \leq \exp\left(-\frac{C^2 D^2 n^{2\epsilon}}{M^2}\right)$$

Proof: see appendix.

By letting $D = 1$ and $\epsilon = 1$, one gets that the probability $P(\|\mathbf{g}\| \geq n^{3/4}) = P(\|\mathbf{g}\|^2 \geq n^{3/2}) = P(\frac{1}{n}\|\mathbf{g}\|^2 \geq n^{1/2})$ is smaller than $e^{-\frac{c^2 n^2}{M^2}}$. This is similar to the Gaussian case, and is sufficient to prove Theorem 10 in the case in which bounded variables are used instead of Gaussian variables.

To summarize the positivity issue, the weight vector α comes out positive due to the fact that it is the leading eigenvector of a matrix whose entries have a positive mean (Propositions 7 and 8). Theorem 10 made the connection between matrices which have the property of a positive mean and the positivity of its leading eigenvector in a probabilistic setting.

4.2 Sparsity

We move our attention to the issue of the sparsity of the weight vector α . It has been observed in the past that the key for sparsity lies in the positive combination of terms (cf. Lee and Seung, 1999) — therefore there is a strong, albeit anecdotal, relationship between the positivity of α and the sparsity feature. Below, we will establish a relationship between the spectral properties of the relevant and the irrelevant feature sets, and the sparsity of the feature vector.

Let M be the (normalized) data matrix consisting of n rows. Assume that the rows of the matrix have been sorted such that the first n_1 rows are relevant features, and the next $n_2 = n - n_1$ features are irrelevant. Let the matrix containing the first n_1 rows be noted as M_1 , and let the matrix containing the rest of the rows be M_2 , i.e, $M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$.

We study the elements of the vector α that correspond to the irrelevant features to show that these elements have a small magnitude. If these n_2 weights are low, we can expect the effect of the

2. This is the inequality one gets while proving Hoeffding’s inequality. It differs from the canonical inequality in that the one sided case has a factor of 2 improvement.

associated features to be small. We will next tie the average of these values to the spectral properties of M_1 and M_2 .

Recall the weight vector α is the first eigenvector of the matrix $G_{ij} = (\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top Q Q^\top \mathbf{m}_j$, where \mathbf{m}_i are the rows of the matrix M , and Q is a matrix containing k orthonormal columns q_i , $i = 1..k$. Let λ be the largest eigenvalue of G .

Lemma 13 (sum of irrelevant features' weight) *Using the above definitions, let $\gamma_i, i = 1..n_2$ be the eigenvalues of $M_2 M_2^\top$.*

$$\sum_{i=n_1+1}^n \alpha_i \leq \sqrt{\frac{\sum_{i=1}^{n_2} \gamma_i^2}{\lambda}} .$$

Proof:

Note that if $\sum_{i=n_1+1}^n \alpha_i \leq 0$ the lemma holds trivially. Let $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ be the vector with n_1 zeros and n_2 ones.

$$\sum_{i=n_1+1}^n \alpha_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top \alpha = \sqrt{\begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top \alpha \alpha^\top \begin{bmatrix} 0 \\ 1 \end{bmatrix}} \leq \sqrt{\frac{\begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top G \begin{bmatrix} 0 \\ 1 \end{bmatrix}}{\lambda}} ,$$

where the last inequality follows from the spectral decomposition of the positive definite matrix G , to which α is an eigenvector with an eigenvalue of λ .

Let \hat{G} be the matrix containing the point-wise squares of the elements of MM^\top , i.e., $\hat{G}_{ij} = (\mathbf{m}_i^\top \mathbf{m}_j)^2$.

Let \hat{Q} be a matrix containing $n - k$ orthonormal columns that span the space orthogonal to Q . $(\hat{G} - G)$ has a structure similar to G , but with \hat{Q} instead of Q , and is also positive definite. To see this notice that $QQ^\top + \hat{Q}\hat{Q}^\top = I_n$ and that the ij element of $(\hat{G} - G)$ is therefore given by

$$\hat{G}_{ij} - G_{ij} = (\mathbf{m}_i^\top \mathbf{m}_j)^2 - (\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top Q Q^\top \mathbf{m}_j = (\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top \hat{Q} \hat{Q}^\top \mathbf{m}_j .$$

We have

$$\begin{aligned} \begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top G \begin{bmatrix} 0 \\ 1 \end{bmatrix} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top \hat{G} \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top (G - \hat{G}) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \leq \\ &\leq \begin{bmatrix} 0 \\ 1 \end{bmatrix}^\top \hat{G} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \|M_2 M_2^\top\|_F^2 = \sum_{i=1}^{n_2} \gamma_i^2 . \end{aligned}$$

The lemma follows. \square

The denominator in the bound $(\sqrt{\lambda})$ is exactly the quantity that our algorithms maximize. The higher this value, the tighter the bound. In the ideal case, almost all of the energy in the features is contained in the space spanned by the columns of Q . Let $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ be the vector of n_1 ones, followed by n_2 zeros. We have:

$$\lambda = \alpha^\top G \alpha \geq \frac{\begin{bmatrix} 1 \\ 0 \end{bmatrix}^\top G \begin{bmatrix} 1 \\ 0 \end{bmatrix}}{n_1} \sim \frac{\begin{bmatrix} 1 \\ 0 \end{bmatrix}^\top \hat{G} \begin{bmatrix} 1 \\ 0 \end{bmatrix}}{n_1} = \frac{\|M_1 M_1^\top\|_F^2}{n_1} .$$

The bound will be tightest if all relevant features have high correlations. In this case, we can expect $\sqrt{\lambda}$ to be linear in n_1 . Therefore the addition of more relevant features reduces the weights of the irrelevant features.

Without any assumption about the entries of the data matrix M , we cannot say much on the numerator of the bound in Lemma 13. However, by using random matrices, we can qualitatively evaluate this bound.

The numerator of the bound contains the term $\sum_{i=1}^{n_2} \gamma_i^2$, which is just the squared Frobenius norm of $M_2 M_2^\top$. Let $W_2 = M_2 M_2^\top$, $\|W_2\|_F^2 = \text{trace}(W_2 W_2^\top) = \text{trace}(W_2^2)$. The expectation of this expression (where M_2 is drawn from a random distribution), normalized by the number of rows in M_2 , is called the *second moment* of W_2 . More generally, if A is a random matrix of size $n \times n$, the k moment of it is defined as $m_k = \frac{1}{n} \text{trace}(A^k)$. For large n this definition coincides with the moments of the distribution of the eigenvalues of the matrix A .

Consider now matrices W of the form $W = \frac{1}{q} M M^\top$, where M is an $n \times q$ random matrix with zero mean (weakly) independent elements with a variance of 1. These matrices are called Wishart matrices. Note that the elements in the matrix M need not be Gaussian. The rows of the matrix M are not explicitly normalized. However, the scale of $\frac{1}{q}$ can be thought of as a scale of $\frac{1}{\sqrt{q}}$ for each element of M , and due to the central limit theorem we can expect for large enough values of q to have the mean of each row approximately zero and the norm of each row approximately 1. Hence, Wishart matrices well approximate our data matrices, if we are willing to assume that the elements of our data matrices are independent. For the bulk of irrelevant features, this may be a reasonable assumption.

For large n , the moments of the Wishart matrices are well approximated by the Narayana polynomials $m_k = \sum_{j=0}^{k-1} \frac{(n/j)^j}{j+1} \binom{k}{j} \binom{k-1}{j}$. In particular, the second moment is given by $1 + n/q$. Since the moment is the appropriate trace scaled by $\frac{1}{n}$, we can expect $\sqrt{\sum_{i=1}^{n_2} \gamma_i^2}$ to behave similarly to $\sqrt{n_2(1 + n_2/q)}$.

Therefore, the rate in which the bound on the sum of squares of weights of irrelevant features grow is mostly linear. The implication is that the $Q - \alpha$ algorithm is robust to many irrelevant features: to a first approximation, the bound on the average squared weight of an irrelevant feature remains mostly the same, as the number of irrelevant features increases.

In Sec. 6 we will present a number of experiments, both with synthetic and real data. Fig. 2(b) shows the weight vector α for a random data matrix M , and for a synthetic experiment (6 relevant features out of 202). One can clearly observe the positivity and sparsity of the recovered weight vector — even for a random matrix.

4.3 Sparsity and Generalization

The sparsity of the returned vector of weights does more than just directly ensure that the irrelevant features are left out; it also helps the generalization ability of the returned kernel by lowering the trace of the kernel matrix.

Recall that in our optimization scheme, the vector of weights α has a norm of 1, and is expected to have all positive elements. For norm-1 vectors, the sum $\sum_i \alpha_i$ is highest when the elements of the vector are evenly distributed. Due to the sparsity of the returned vector of weights, we can expect the above sum to be much lower than what we get with a uniform weighting of the data.

Consider the matrix A_α , the linear kernel matrix based on the weights returned by the $Q - \alpha$ algorithm. A_α , which equals $\sum \alpha_i m_i m_i^\top$, is a weighted sum of rank-one matrices. Since our features are normalized to have norm-1, each such rank-one matrix $m_i m_i^\top$ has a trace of 1. Therefore, the trace of the kernel matrix A_α is exactly the sum of the elements of the vector α .

From the discussion above, the trace of the kernel matrix returned by the $Q - \alpha$ algorithm is expected to be low. This is exactly the criteria for a good kernel matrix expressed by “the trace bounds.” The trace bounds are Rademacher complexity type of error bounds for classifiers that are linear combinations of kernel functions (Bousquet and Herrmann, 2003). These bounds relate the trace of the kernel matrix used for training with the generalization error of the classifiers that were trained. The lower the trace, the lower the bound on the difference between the testing error and the training error.

Although there is no immediate analog to the concept of generalization error in the unsupervised case, we can expect a similar criteria to hold for this case as well. A good kernel matrix for unsupervised learning should support the separation given by the set of true underlying labels (although unknown). It should not, however, support any random labeling. This is exactly what is measured by the Rademacher process: how well does the class of functions used for learning separate random partitions of the training set.

In supervised learning, feature selection can be a major cause of over fitting. Consider the common case where the number of features is much larger than the number of examples. In this case, it might be possible to construct a classifier with a very low training error, which employs only few selected features. This reliance on a small portion on the data when making the classification leads to poor generalization performance. This problem was pointed out, for example, by Long and Vega (2003), who suggested, in their simplest and most effective solution (“AdaBoost-NR”), to encourage redundancy in the pool of participating features. The $Q - \alpha$ algorithm, as a result of optimizing the cost function subject to the constraint that the norm of the α vector is one, has a similar property. It prefers to divide high weights between a group of correlated features, rather than to pick one promising feature out of this group and assign it a higher weight.

5. Representing Higher-order Cumulants using Kernel Methods

The information on which the $Q - \alpha$ method relies on to select features is contained in the matrix G . Recall that the criterion function underlying the $Q - \alpha$ algorithm is a sum over all pairwise feature vector relations:

$$\text{trace}(Q^\top A_\alpha^\top A_\alpha Q) = \alpha^\top G \alpha,$$

where G is defined such that $G_{ij} = (\mathbf{m}_i^\top \mathbf{m}_j) \mathbf{m}_i^\top Q Q^\top \mathbf{m}_j$. It is apparent that feature vectors interact in pairs and the interaction is *bilinear*. Consequently, cumulants of the original data matrix M which are of higher order than two are not being considered by the feature selection scheme. For example, if M were to be decorrelated (i.e., MM^\top is diagonal) the matrix G would be diagonal and the feature selection scheme would select only a single feature rather than a feature subset.

In this section we employ the so called “kernel trick” to allow for cumulants of higher orders among the feature vectors to be included in the feature selection process. Kernel methods in general have been attracting much attention in the machine learning literature — initially with the introduction of the support vector machines (Vapnik, 1998) and later took a life of their own (see Scholkopf and Smola, 2002). The common principle of kernel methods is to construct nonlinear variants of linear algorithms by substituting inner-products by nonlinear kernel functions. Under certain conditions this process can be interpreted as mapping of the original measurement vectors (so called “input space”) onto some higher dimensional space (possibly infinitely high) commonly referred to as the “feature space” (which for this work is an unsuccessful choice of terminology since the word “feature” has a different meaning). Mathematically, the kernel approach is defined as follows: let

$\mathbf{x}_1, \dots, \mathbf{x}_l$ be vectors in the input space, say R^q , and consider a mapping $\phi(\mathbf{x}) : R^q \rightarrow \mathcal{F}$ where \mathcal{F} is an inner-product space. The kernel-trick is to calculate the inner-product in \mathcal{F} using a kernel function $k : R^q \times R^q \rightarrow R$, $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, while avoiding explicit mappings (evaluation of) $\phi(\cdot)$. Common choices of kernel selection include the d 'th order polynomial kernels $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d$ and the Gaussian RBF kernels $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2)$. If an algorithm can be restated such that the input vectors appear in terms of inner-products only, one can substitute the inner-products by such a kernel function. The resulting kernel algorithm can be interpreted as running the original algorithm on the space \mathcal{F} of mapped objects $\phi(\mathbf{x})$. Kernel methods have been applied to the support vector machine (SVM), principal component analysis (PCA), ridge regression, canonical correlation analysis (CCA), QR factorization and the list goes on. We will focus below on deriving a kernel method for the $Q - \alpha$ algorithm.

5.1 Kernel $Q - \alpha$

We will consider mapping the rows \mathbf{m}_i^\top of the data matrix M such that the rows of the mapped data matrix become $\phi(\mathbf{m}_1)^\top, \dots, \phi(\mathbf{m}_n)^\top$. Since the entries of G consist of inner-products between pairs of mapped feature vectors, the interaction will be no longer bilinear and will contain higher-order cumulants whose nature depends on the choice of the kernel function.

Replacing the rows of M with their mapped version introduces some challenges before we could apply the kernel trick. The affinity matrix $A_\alpha = \sum_i \alpha_i \phi(\mathbf{m}_i) \phi(\mathbf{m}_i)^\top$ cannot be explicitly evaluated because A_α is defined by *outer-products* rather than inner-products of the mapped feature vectors $\phi(\mathbf{m}_i)$. The matrix Q holding the eigenvectors of A_α cannot be explicitly evaluated as well and likewise the matrix $Z = A_\alpha Q$ (in step 4). As a result, kernelizing the $Q - \alpha$ algorithm requires one to represent α without explicitly representing A_α and Q both of which were instrumental in the original algorithm. Moreover, the introduction of the kernel should be done in such a manner to preserve the key property of the original $Q - \alpha$ algorithm of producing a sparse solution.

Let $V = MM^\top$ be the $n \times n$ matrix whose entries are evaluated using the kernel $v_{ij} = k(\mathbf{m}_i, \mathbf{m}_j)$. Let $Q = M^\top E$ for some $n \times k$ (recall k being the number of clusters in the data) matrix E . Let $D_\alpha = \text{diag}(\alpha_1, \dots, \alpha_n)$ and thus $A_\alpha = M^\top D_\alpha M$ and $Z = A_\alpha Q = M^\top D_\alpha V E$. The matrix Z cannot be explicitly evaluated but $Z^\top Z = E^\top V D_\alpha V D_\alpha V E$ can be evaluated. The matrix G can be expressed with regard to E instead of Q :

$$\begin{aligned}
 G_{ij} &= (\phi(\mathbf{m}_i)^\top \phi(\mathbf{m}_j)) \phi(\mathbf{m}_i)^\top Q Q^\top \phi(\mathbf{m}_j) \\
 &= k(\mathbf{m}_i, \mathbf{m}_j) \phi(\mathbf{m}_i)^\top (M^\top E) (M^\top E)^\top \phi(\mathbf{m}_j) \\
 &= k(\mathbf{m}_i, \mathbf{m}_j) \mathbf{v}_i^\top E E^\top \mathbf{v}_j
 \end{aligned}$$

where $\mathbf{v}_1, \dots, \mathbf{v}_n$ are the columns of V . Step 5 of the $Q - \alpha$ algorithm consists of a QR factorization of Z . Although Z is uncomputable it is possible to compute R and R^{-1} directly from the entries of $Z^\top Z$ without computing Q using the Kernel Gram-Schmidt described by Wolf and Shashua (2003). Since $Q = ZR^{-1} = M^\top D_\alpha V E R^{-1}$ the update step is simply to replace E with $E R^{-1}$ and start the cycle again. In other words, rather than updating Q we update E and from E we obtain G and from there the newly updated α . The kernel $Q - \alpha$ is summarized below:

Definition 14 (Kernel $Q - \alpha$) *Let M be an uncomputable matrix with rows $\phi(\mathbf{m}_1)^\top, \dots, \phi(\mathbf{m}_n)^\top$ where $\phi(\cdot) : R^n \rightarrow \mathcal{F}$ is a mapping from input space to a feature space and which is endowed with a kernel function $\phi(\mathbf{m}_i)^\top \phi(\mathbf{m}_j) = k(\mathbf{m}_i, \mathbf{m}_j)$. Therefore the matrix $V = MM^\top$ is a computable*

$n \times n$ matrix. Let $E^{(0)}$ be an $n \times k$ matrix selected such that $M^\top E^{(0)}$ has orthonormal columns. Perform the following steps through a cycle of iterations with index $r = 1, 2, \dots$

1. Let $G^{(r)}$ be a $n \times n$ matrix whose (i, j) components are $k(\mathbf{m}_i, \mathbf{m}_j) \mathbf{v}_i^\top E^{(r-1)} E^{(r-1)\top} \mathbf{v}_j$.
2. Let $\alpha^{(r)}$ be the largest eigenvector of $G^{(r)}$, and let $D^{(r)} = \text{diag}(\alpha_1^{(r)}, \dots, \alpha_n^{(r)})$.
3. Let $Z^{(r)}$ be an uncomputable matrix

$$Z^{(r)} = (M^\top D^{(r)} M)(M^\top E^{(r-1)}) = M^\top D^{(r)} V E^{(r-1)}.$$

Note that $Z^{(r)\top} Z^{(r)}$ is a computable $k \times k$ matrix.

4. $Z^{(r)} \xrightarrow{QR} QR$. It is possible to compute directly R, R^{-1} from the entries of $Z^{(r)\top} Z^{(r)}$ without explicitly computing the matrix Q (see (Wolf and Shashua, 2003)).
5. Let $E^{(r)} = E^{(r-1)} R^{-1}$.
6. Increment index r and go to step 1.

The result of the algorithm is the weight vector α and the design matrix G which contains all the data about the features.

6. Experiments

We have validated the effectiveness of the proposed algorithms on a variety of data sets. Our main focus in the experiments below is in the unsupervised domain, which has received much less attention in the feature selection literature than the supervised one.

SYNTHETIC DATA

We compared the $Q - \alpha$ algorithm with three classical filter methods (Pearson correlation coefficients, Fisher criterion score and the Kolmogorov-Smirnoff test), standard SVM and the wrapper method using SVM of Weston et al. (2001). The data set we used follow precisely the one described by Weston et al., which was designed for supervised 2-class inference. Two experiments were designed, one with 6 relevant features out of 202 referred to as “linear” problem, and the other experiment with 2 relevant features out of 52 designed in a more complex manner and referred to as “non-linear” problem. In the linear data the class label $y \in \{-1, 1\}$ was drawn at equal probability. The first six features were drawn as $x_i = yN(i, 1)$, $i = 1..3$, and $x_j = N(0, 1)$, $j = 4..6$ at probability 0.7, otherwise they were drawn as $x_i = N(0, 1)$, $i = 1..3$, and $x_j = yN(i - 3, 1)$, $j = 4..6$. The remaining 196 dimensions were drawn from $N(0, 20)$. The reader is referred to (Weston et al., 2001) for details of the non-linear experiment. We ran $Q - \alpha$ on the two problems once with known classes (supervised version) and with unknown class labels (unsupervised version). In the supervised case the selected features were used to train an SVM and in the unsupervised case the class labels were not used for the $Q - \alpha$ feature selection but were used for the SVM training. The unsupervised test appears artificial but is important for appreciating the strength of the approach as the results of the unsupervised are only slightly inferior to the supervised test. For each size of training set we report the average test error on 500 samples over 30 runs. In Fig. 3(a) we *overlay* the $Q - \alpha$ results (prediction error of the SVM on a testing set) on the figure obtained by Weston et al.. The performance of the supervised $Q - \alpha$ closely agrees with the performance of the wrapper

SVM feature selection algorithms. The performance of the unsupervised version does not fall much behind.

Since our method can handle more than two classes we investigated the scaling-up capabilities of the algorithm as we increase the number of classes in an unsupervised setting. For $n_c = 2, 3, \dots$ classes we sampled n_c cluster centers in 5D space (5 coordinates per center) in the 5D cube where each cluster center coordinate is uniformly sampled in the interval $[-1, 1]$. For each cluster we also uniformly samples a diagonal covariance matrix with elements taken from the interval $[0, .02]$. Around each of the n_c class centers we sampled $\lceil \frac{60}{n_c} \rceil$ points according to a normal distribution whose mean is the class center and with a the random covariance matrix. We added 120 additional coordinates drawn similarly around n_c centers sampled uniformly inside the 120D hypercube with edges of length 2, according to the same rules. Each such added coordinate was permuted by a random permutation to break the correlation between the dimensions. Thus each of the 60 points lives in a 125-dimensional space out of which only the first five dimensions are relevant. We ran the $Q - \alpha$ algorithm on the data matrix and obtained the weight vector α and computed the sparsity gap - i.e the ratio between the average weight of the first five features and the average weight of the rest 120 features. Ideally the ratio should be high if the algorithm consistently succeeds in selecting the first three coordinates as the relevant ones.

Fig. 3(b) illustrates the results of this experiment in a graph whose x -axis runs over the number of classes k and the y -axis displays the sparsity gap (the ratio discussed above). Each experiment was repeated 20 times and the results in the plot are the average of the 20 runs and the 25 and 75 percentiles. In general the error bars for small number of classes are large indicating that some experiments are much more difficult than others. This is probably a results of the cluster centers being close to one another in some of the experiments.

There are three plots on the graph. The solid blue describes the result obtained when choosing $k = n_c$. For small number of classes this gives the best results. The dashed green plot describes the results obtained while choosing $k = n_c + 2$. This choice seems to result with a smaller variance between experiments. The explanation might be that variance is a results of the fact that in some experiments the cluster centers are close, making the separation difficult. Taking a large value of k captures more complex details about the cluster structure. For example: when two clusters have close centers the resulting distribution might look like one strong cluster in the middle, and some cluster tails around it. The red plot is the one obtained when under estimating the number of clusters and taking $k = \max(1, n_c - 2)$. This has the largest variance, but the best (in average) when the number of clusters is large. The reason might be that focusing on the clusters which are well separated is better than trying to capture information from all clusters. This is however, a “risky” strategy leading to a large variance.

One can see that the algorithm performed well until $k = 6$. After that the sparsity ratio is still larger than one most of the time, but separation is not easy. It is possible to get better performance in average by underestimating k by more than 2 at the price of a higher variance. Good performance up to 6 clusters and a sparsity gap around 5 – 10 are not “magical numbers”. For other feature selection problems (e.g., a different number of points per cluster, other sampling probabilities, etc.) we can get good performance for more classes or for less depending on the complexity of the problem.

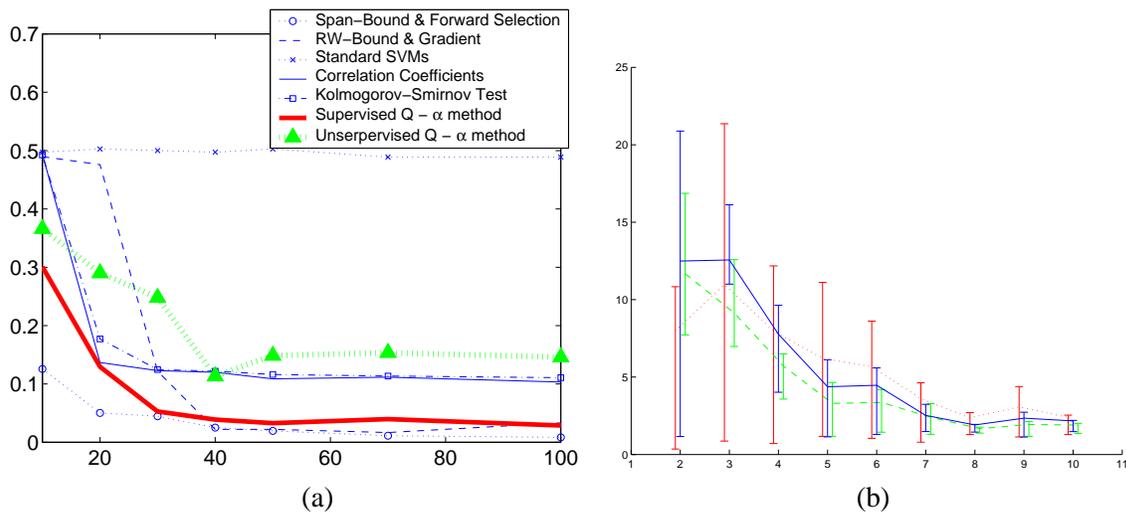


Figure 3: (a) Comparison of feature selection methods following (Weston et al., 2001). Performance curves of $Q - \alpha$ were overlaid on the figure adapted from (Weston et al., 2001). The x -axis is the number of training points and the y -axis is the test error as a fraction of test points. The thick solid lines correspond to the $Q - \alpha$ supervised and unsupervised methods (see text for details). (b) Performance of a test with five relevant features and 120 irrelevant ones with n_c clusters represented by the x -axis of the graph. The y -axis represents the sparsity gap (see text for details). The three graphs are solid blue for $k = n_c$, dashed green for $k = n_c + 2$ and dotted red for $k = \max(1, n_c - 2)$. One can see that the unsupervised $Q - \alpha$ sustained good performance up to 6 classes in this settings.

REAL IMAGE UNSUPERVISED FEATURE SELECTION

The strength of the $Q - \alpha$ method is that it applies for unsupervised settings as well as supervised. An interesting unsupervised feature selection problem in the context of visual processing is the one of automatic selection of relevant features which discriminate among perceptual classes. Assume one is given a collection of images where some of them contain pictures of a certain object class (say, green frogs, the *Rana clamitans* specie) and other images contain pictures of a different class of objects (say, American toads) — see Fig. 4. We would like to automatically, in an unsupervised manner, select the relevant features such that a new picture could be classified to the correct class membership.

The features were computed by matching patches of equal size of 20×20 pixels in the following manner. Assuming that the object of interest lies in the vicinity of the image center, we defined 9 “template” patches arranged in a 3×3 block centered at the image. For example, in one experiment, we had 27 images (18 from one class and 9 from the other), which in turn defines $27 \times 9 = 243$ feature coordinates. Each image was sampled by 49 “candidate” patches (covering the entire image) where each of the 243 template patches was matched against the 49 patches in its respective image and the score of the best match was recorded in 243×27 data matrix. The matching between a pair of patches was based on L_1 -distance between the respective color histograms in HSV space. We ran the $Q - \alpha$ algorithm with $k = 2$. The resulting α weight vector forms a feature selection from which we create a submatrix of data points and construct its affinity matrix and the associated matrix of eigenvectors Q . The rows of the Q matrix were clustered using k-means into two clusters.

This experiment was done in an unsupervised settings. As a measure of performance we used the percent of samples with labels matching the correct labeling (the maximum over the two flips of the class labels). Performance varied between 80% to 90% correct assignments over many experiments over several object classes (including elephants, sea elephants, and so forth). Images were taken from CalPhotos: Animals (<http://elib.cs.berkeley.edu/photos/fauna/>). For each class we took all images in which the animal appears, e.g., we removed all tadpoles images from the green frog class. This performance was compared to spectral clustering using all the features (243 in the above examples) which provided a range of 55% to 65% correct classification.

Fig. 5(a) and Fig. 5(b) show the 20 most relevant templates selected for the two classes, and Fig. 5(c) shows the alpha values. Note that the α weights are positive as predicted from Theorem 10 and that only few of the features have very high weights.

KERNEL $Q - \alpha$ EXPERIMENTS

One of the possible scenarios for which a polynomial (for example) kernel is useful is when hidden variables affect the original feature measurements and thus create non-linear interactions among the feature vectors. We consider the situation in which the original measurement matrix M is multiplied, element wise, with a hidden variable matrix whose entries are ± 1 . The value of the hidden state was changed randomly every 8 measurements and independently for each feature. This scheme simulates measurements taken in “sessions” where a session lasts for 8 sample data points. As a result, the expectation of the inner product between any two feature vectors is zero yet any two feature vectors contain higher-order interactions which could come to bear using a polynomial kernel.

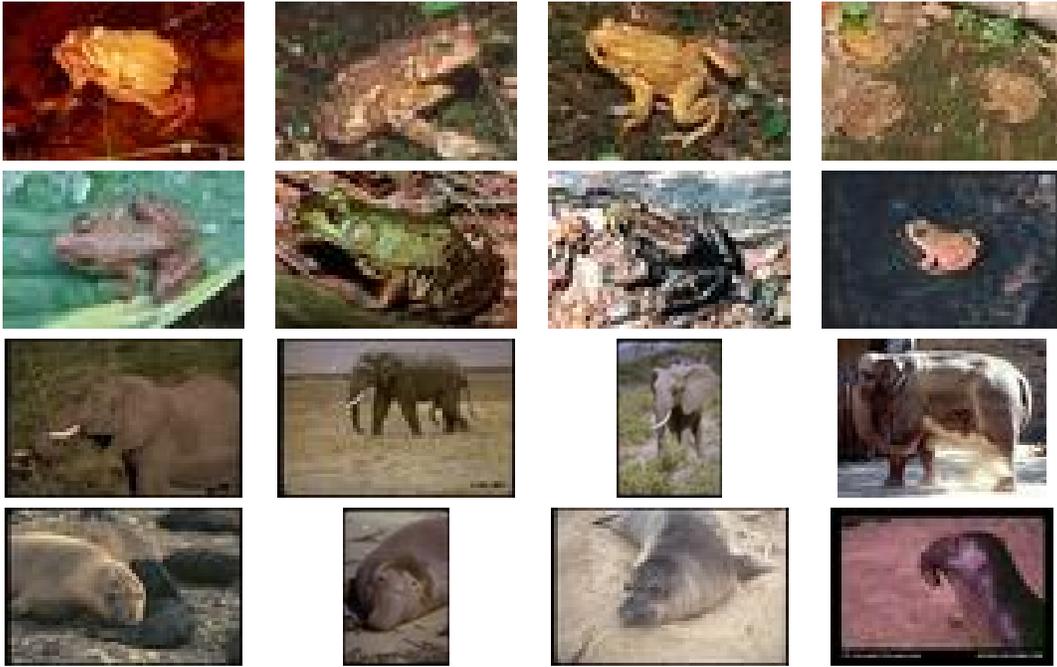


Figure 4: Image samples of several animal classes — American toad (top row) and Green frogs (*Rana clamitans*), elephants, and sea elephants. The objects appear in various positions, illumination, context and size.

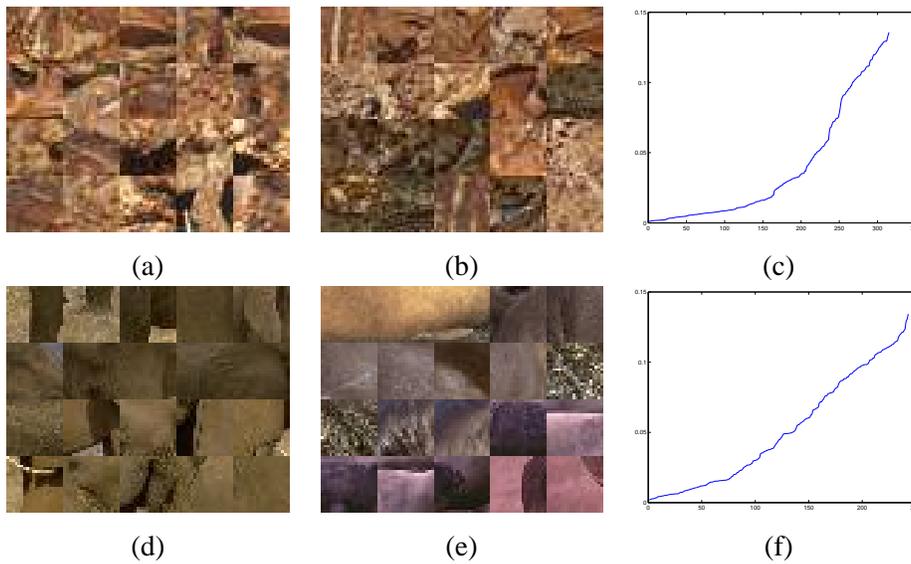


Figure 5: Unsupervised feature selection for automatic object discrimination from images. (a),(b) the first 20 features from pictures containing the American frog and the Green frog ranked by the α weight vector. (c) the (sorted) α values. (d),(e),(f) similar to the elephant and sea elephant.

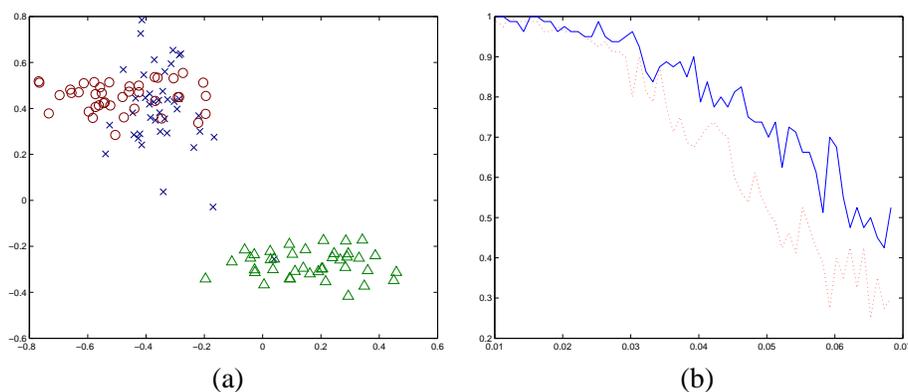


Figure 6: (a) 2D slice out of the relevant features in the original data matrix used in the synthetic experiment, showing three clusters. (b) A graph showing the success rate for the 2nd order polynomial kernel (solid blue), and for a preprocessing of the data (dashed red). The results are shown over the parameter λ specifying the variance of the original data set (see text). The success rate of the regular $Q - \alpha$ algorithm was constantly zero and is not shown.

The kernel we used in this experiment was a sum of second-order polynomial kernels each over a portion of 8 entries of the feature vector:

$$k(\mathbf{m}_i, \mathbf{m}_j) = \sum_k (m_i^k m_j^k)^2,$$

where m_i^k represents the k 'th section of 8 successive entries of the feature vector \mathbf{m}_i . The original data was composed out 120 sample points with 60 coordinates out of which 12 were relevant and 48 were irrelevant. The relevant features were generated from three clusters, each containing 40 points. The points of a cluster were Normally distributed with a mean vector drawn uniformly from the unit hypercube in \mathcal{R}^{12} and with a diagonal covariance matrix with entries uniformly distributed in the range $[\lambda, 2\lambda]$, where λ is a parameter of the experiment. A 2D slice out of the relevant 12 dimensions is shown in figure 6(a). The irrelevant features were generated in a similar manner, where for each irrelevant feature the sample points were permuted independently in order to break the interactions between the irrelevant features. This way it is impossible to distinguish between a single relevant feature and a single irrelevant feature.

We considered an experiment to be successful if among the 12 features with the highest α values, at least 10 were from the relevant features subset. The graph in figure 6(b) shows the success rate for the kernel $Q - \alpha$ algorithm averaged over 80 runs. It also shows, for comparison, the success rate for experiments conducted by taking the square of every element in the measurements matrix followed by running the original $Q - \alpha$ algorithm. The success rate for the original $Q - \alpha$ algorithm on the unprocessed measurements was constantly zero and is not shown in the graph.

GENOMICS

Synthetic data We have tested our algorithm against the synthetic model of gene expression data (“microarrays”) given in (Ben-Dor et al., 2001). This synthetic model has 6 parameters m, a, b, e, d, s ,

explained below. a samples are drawn from class A , and b samples are drawn from class B . Each sample has m dimensions - em samples are drawn randomly using the distribution $N(0, s)$. The rest of the $(1 - e)m$ features are drawn using either $N(\mu_A, \mu_{As})$ or $N(\mu_B, \mu_{Bs})$, depending on the class of the sample. The means of the distributions μ_A and μ_B are uniformly chosen from the interval $[-1.5d, 1.5d]$.

In (Ben-Dor et al., 2001) the parameters of the model were estimated to best fit the gene expressions of the leukemia data set: $m = 600, a = 25, b = 47, e = 0.72, d = 555, s = 0.75$ ³. Similarly to (Ben-Dor et al., 2001), we varied one of the parameters m, d, e, s while fixing the other parameters to the values specified above. This enabled us to compare the performance of the $Q - \alpha$ algorithm to the performance of their Max-Surprise algorithm (MSA).

Our algorithm was completely robust to the number of features m . It always chose the correct features using as few as 5 features. MSA needed at least 250 features, since it used the redundancy in the features in order to locate the informative features. Both algorithms are invariant to the distance between the means of the distributions determined by d , and perform well for $d \in [1, 1000]$. The percentage of irrelevant features, e , can reach 95% for MSA and 99.5% for our algorithm. Such performance suggests that the data set is not very difficult.

The parameter s effects the spread of each class. While MSA was able to handle values of s reaching 2, our algorithm was robust to s , and was at least 30 times more likely to choose a relevant feature than an irrelevant one, even for $s > 1000$.

Real genomics data sets We evaluated the performance of the $Q - \alpha$ algorithm for the problem of gene selection on four data sets containing treatment outcome or status studies (see Wolf et al., 2005, for the full report). The first was a study of treatment outcome of patients with diffuse large cell lymphoma (DLCL), referred to as “lymphoma” (Shipp et al., 2002). The dimensionality of this data set was 7,129 and there were 32 samples with good successful outcome and 26 with unsuccessful outcome. The second was a study of treatment outcome of patients with childhood medulloblastomas (Pomeroy et al., 2002), referred to as “brain”. The dimensionality of this data set was 7,129 and there were 39 samples with good successful outcome and 21 with unsuccessful outcome. The third was a study of the metastasis status of patients with breast tumors (van ’t Veer et al., 2002), referred to as “breast met”. The dimensionality of this data set was 24,624 and there were 44 samples where the patients were disease free for 5 years after onset and 34 samples where the tumors metastasized within five years. The fourth is an unpublished study of breast tumors (Ramaswamy) for which corresponding lymph nodes either were cancerous or not, referred to as “lymph status”. The dimensionality of this data set is 12,600 with 47 samples positive for lymph status and 43 negative for lymph status.

For the four data sets with label information classification accuracy was used as a measure of the goodness of the (unsupervised) $Q - \alpha$ algorithm. We compared the leave-one-out error on these data sets with that achieved by both supervised and unsupervised methods of gene selection. The supervised methods used were signal-to-noise (SNR) (Golub et al., 1999), radius-margin bounds (RMB) (Chapelle et al., 2002; Weston et al., 2001), and recursive feature elimination (RFE) (Guyon et al., 2002). The unsupervised methods used were PCA and gene shaving (GS) (Hall, 2000). In the unsupervised mode the class labels were ignored — and thus in general one should expect the supervised approaches to produce superior results than the unsupervised ones. A linear support

3. The leukemia data set has over 7000 gene expressions but contains much redundancy. Ben-Dor et al. (2001) estimated the effective number of features to be 600 and we follow their choice parameters to allow comparison. Note below that the problem becomes easier as the number of features increase as long as the ratio of relevant features is fixed

vector machine classifier was used for all the gene selection methods. Parameters for SNR, RFE, and RMB were chosen to minimize the leave-one-out error. For the $Q - \alpha$ algorithm we took $k = 6$ for all experiments, to allow for more complex structures than just two clusters. For the breast me data set and for the lymph status data set we took only the first 7,000 features to reduce the computation complexity.

A summary of the results appear in table 1. The $Q - \alpha$ algorithm considerably out-performs all other unsupervised methods. Furthermore, and somewhat intriguing, is that the $Q - \alpha$ algorithm is competitive with the other supervised algorithm (despite the fact that the labels were not taken into account in the course of running the algorithm) and performs *significantly better* on the lymph status of breast tumors as compared to all other gene selection approaches — including the supervised methods.

Method	brain	lymph status ¹	breast met. ¹	lymp-homa
RAW	32	44	34	27
PCA5	22	47	33	40
PCA10	26	47	26	27
PCA20	25	47	25	29
PCA30	31	47	31	33
PCA40	31	47	31	33
PCA50	30	47	30	33
GS5	20	45	32	33
GS10	24	43	31	30
GS20	28	47	32	31
GS30	30	44	33	33
$Q - \alpha$	15	19	22	15
SNR	16	42	29	18
RFE	14	38	26	14
RMB	13	39	24	14

Table 1: The table entries show the Leave-one-out classification errors for the supervised and unsupervised algorithms on the various data sets. In both PCAN and GSN the number N the number of components used. ¹ Only the first 7,000 genes were used.

7. Conclusions

In this work we presented an algebraic approach to variable weighting, which is based on maximizing a score based on the spectral properties of the kernel matrix. The approach has the advantage of being suitable to unsupervised feature selection, but can also be applied in the supervised settings.

It is interesting to compare the algebraic approach presented in this work to probabilistic approaches which take a "holistic" view of the data such as the information bottleneck (Tishby et al., 1999) and the infomax (Linsker, 1988; Vasconcelos, 2003). The gap that exists between the algebraic and the probabilistic tools of machine learning make a direct comparison to information-based feature selection criteria a subject for future work. However, it is evident that algebraic meth-

ods have the advantages of not requiring the estimation of probability distributions, of being more suitable for application on continuous data and, in general, for being easier to optimize for. We conducted a limited experimental comparison to an information-bottleneck method called sufficient dimensionality-reduction (Shashua and Wolf, 2004), and more work is required.

The emergence of sparsity and positiveness in our simple least square optimization function, is a surprising result, that might indicated the possibility of similar results in other algebraic methods of machine learning. For example, it might be interesting to examine if the vector of examples' weights returned by the regularized least squares classification method (Rifkin et al., 2003) would be considered sparse by our definition of sparseness. Regularized least squares method are similar to Support Vector Machines in many ways, only SVMs are known to produce sparse solutions.

As a last remark, we would like to point out that the methods presented in this work are extremely flexible and can be extended. For example, to the case of semi-supervised learning (Shashua and Wolf, 2004).

Acknowledgments

We would like to thank anonymous JMLR reviewers for many useful suggestions. We would like to thank Ofer Zeitouni, Michael Ben-Or and Alex Samorodnitsky for assistance with the proof of Theorem 1. We would like to thank Sayan Mukherjee for providing many of the experimental results on the genomics data sets.

A. S. thanks the financial contribution from Israeli Science Foundation grant No. 189/03 and contribution from the German Israel Foundation grant No. 773/2004.

Appendix A. Positivity of α

The proofs for the claims and theorems made in Section 7 are presented below.

Proposition 7 *The minimal value of $f = (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{c})$ where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in R^q$ are defined over the unit hypersphere is $-1/8$.*

Proof: The QR decomposition of 3 points on the unit hypersphere takes the form:

$$[\mathbf{a}, \mathbf{b}, \mathbf{c}] = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} 1 & \cos(\beta) & \cos(\gamma_1) \\ 0 & \sin(\beta) & \sin(\gamma_1)\cos(\gamma_2) \\ 0 & 0 & \sin(\gamma_1)\sin(\gamma_2) \end{bmatrix} \quad (7)$$

where $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \in R^n$ are three orthogonal vectors.

The problem, therefore, becomes the problem of minimizing

$$f = \cos(\beta)\cos(\gamma_1) (\cos(\beta)\cos(\gamma_1) + \sin(\beta)\sin(\gamma_1)\cos(\gamma_2)) \quad (8)$$

with respect to $\beta, \gamma_1, \gamma_2$. Since γ_2 appears only in the $\cos(\gamma_2)$ expression, it can take only the values of 1 or -1 at the minimum energy point. By symmetry we can assume it to be -1, and the problem reduces to the problem of minimizing $1/2\cos(\beta + \gamma_1)(\cos(\beta + \gamma_1) + \cos(\beta - \gamma_1))$. The minimum occurs when $\cos(\beta - \gamma_1)$ is either 1 or -1. Both problems $1/2\cos(u)(\cos(u) - 1)$ and $1/2\cos(u)(\cos(u) + 1)$ have a minimum of $-1/8$. \square

Proposition 8 *The expected value of $f = (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{c})$ where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathfrak{R}^q$ and \mathbf{c} is uniformly sampled over the unit hypersphere is $(1/q)(\mathbf{a}^\top \mathbf{b})^2$*

Proof: This expectation is given by the following integral

$$\int (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c})(\mathbf{c}^\top \mathbf{b}) d\sigma(\mathbf{c}) = (\mathbf{a}^\top \mathbf{b}) \mathbf{a}^\top \left(\int \mathbf{c} \mathbf{c}^\top d\sigma(\mathbf{c}) \right) \mathbf{b}.$$

\mathbf{c} is taken from a uniform probability and in particular from a symmetric probability, i.e., where the probability of \mathbf{c} and of remains the same under sign flipping of any subset of its entries (e.g., $p(\sqrt{2}[.5, .5, 0, 0]) = p(\sqrt{2}[-.5, .5, 0, 0])$). Therefore, $\int \mathbf{c} \mathbf{c}^\top d\sigma(\mathbf{c})$ is a multiplication of the identity matrix. From linearity of the trace and from the equality $\text{trace}(\mathbf{c} \mathbf{c}^\top) = \mathbf{c}^\top \mathbf{c}$ the trace of this matrix is $\int \mathbf{c}^\top \mathbf{c} d\sigma(\mathbf{c}) = 1$. The matrix $\int \mathbf{c} \mathbf{c}^\top d\sigma(\mathbf{c})$, therefore, is $1/q$ times the identity matrix in \mathfrak{R}^q . The expectation $\int (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c})(\mathbf{c}^\top \mathbf{b}) d\sigma(\mathbf{c})$ then equals $(1/q)(\mathbf{a}^\top \mathbf{b})^2$. \square

Proposition 9 *The expected value of $f = \sum_{i=1}^k (\mathbf{a}^\top \mathbf{b})(\mathbf{a}^\top \mathbf{c}_i)(\mathbf{b}^\top \mathbf{c}_i)$ where $\mathbf{a}, \mathbf{b} \in \mathfrak{R}^q$ and \mathbf{c}_i are orthonormal vectors uniformly sampled over the unit hypersphere in \mathfrak{R}^q is $(k/q)(\mathbf{a}^\top \mathbf{b})^2$.*

Proof: This expectation is given by the following integral

$$(\mathbf{a}^\top \mathbf{b}) \mathbf{a}^\top \left(\sum_{i=1}^k \int \mathbf{c}_i \mathbf{c}_i^\top d\sigma(\mathbf{c}_i | \mathbf{c}_1 \dots \mathbf{c}_{i-1}) \right) \mathbf{b},$$

where the main difference from the proof of Prop. 8 is that now the probability distribution of \mathbf{c}_i is dependent on all the previous $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{i-1}$. Nevertheless, if \mathbf{c}_i are uniformly sampled subject to the orthogonality constraint, the sum of integrals $J = \sum_{i=1}^k \int \mathbf{c}_i \mathbf{c}_i^\top d\sigma(\mathbf{c}_i | \mathbf{c}_1 \dots \mathbf{c}_{i-1})$ is a product over the identity matrix in \mathfrak{R}^q . To see this, consider products of the form $v^\top J v$. From symmetry this product must be the same for every $v \in \mathfrak{R}^q$. i.e, since $v^\top J v$ depends only on dot products (the distribution $d\sigma(\mathbf{c}_i | \mathbf{c}_1 \dots \mathbf{c}_{i-1})$ is a uniform distribution subject to constraints on dot products), it is invariant to a unitary transformation; in particular since any vector can be rotated to any other vector we get that it is not dependent on v . We have $\text{trace}(J) = k$ satisfying the proposition, as $\text{trace}(\sum_{i=1}^k \int \mathbf{c}_i \mathbf{c}_i^\top d\sigma(\mathbf{c}_i | \mathbf{c}_1 \dots \mathbf{c}_{i-1})) = \sum_{i=1}^k \int \mathbf{c}_i^\top \mathbf{c}_i d\sigma(\mathbf{c}_i | \mathbf{c}_1 \dots \mathbf{c}_{i-1}) = k$. \square

Theorem 10 (Probabilistic Perron-Frobenius) *Let $G = g_{ij}$ be a real symmetric $n \times n$ matrix whose entries for $i \geq j$ are independent identically and normally distributed random variables with mean $\mu > 0$ and variance σ^2 . Then, for any $\varepsilon > 0$ there exist n_0 such that for all $n > n_0$ the leading eigenvector \mathbf{v} of G is positive with probability of at least $1 - \varepsilon$.*

Preliminaries: Let $G = \mu J + \sigma S$ where $J = \mathbf{1} \mathbf{1}^\top$ and S_{ij} are i.i.d. sampled according to $N(0, 1)$. Let $\mathbf{e} = \frac{1}{\sqrt{n}} \mathbf{1}$. and let $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ and $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ be the spectrum of G . From the semicircle law (Wigner, 1958) and from (Furedi and Komlos, 1981) it is known that $\lambda_i = \Theta(\sqrt{n})$ for $i = 2, 3, \dots, n$.

The following auxiliary claims would be useful for proving the main theorem.

Lemma 15 (Bounds on Leading Eigenvalue) *Under the conditions of Theorem 10 above, with probability $1 - o(1)$ the leading eigenvalue λ of G falls into the following interval:*

$$\mu n - \Theta(1) \leq \lambda \leq \mu n + \Theta(\sqrt{n}).$$

Proof: From the definition of the leading eigenvalue we have:

$$\begin{aligned}\lambda &= \max_{\|\mathbf{x}\|=1} \mathbf{x}^\top G \mathbf{x} = \mu \left(\sum_i x_i \right)^2 + \sigma \max_{\|\mathbf{x}\|=1} \mathbf{x}^\top S \mathbf{x} \\ &\leq \mu n + \Theta(\sqrt{n})\end{aligned}$$

where from the semicircle law $\max_{\|\mathbf{x}\|=1} \mathbf{x}^\top S \mathbf{x} = \Theta(\sqrt{n})$ and from Cauchy-Schwartz inequality $(\sum_i x_i)^2 \leq n(\sum_i x_i^2) = n$. The lower bound follows from:

$$\begin{aligned}\lambda &\geq \mathbf{e}^\top G \mathbf{e} = \mu n + \sigma \mathbf{e}^\top S \mathbf{e} \\ &= \mu n + \sum_{i,j} S_{ij} / n \geq \mu n - \Theta(1)\end{aligned}$$

Lemma 16 *Under the conditions of Theorem 10 above, with probability $1 - o(1)$ we have the following bound:*

$$\sum_i v_i \geq \sqrt{n} - c \tag{9}$$

for some constant c where v_i are the entries of the leading eigenvector \mathbf{v} of G .

Proof: Let $\mathbf{e} = a\mathbf{v} + \sum_{i=2}^n a_i \mathbf{v}_i$. Since the eigenvectors and \mathbf{e} are of unit norm we have $a^2 + \sum_{i=2}^n a_i^2 = 1$ and without loss of generality we can assume $a > 0$. We have therefore $\mathbf{e}^\top G \mathbf{e} = a^2 \lambda + \sum_i \lambda_i a_i^2$. Since $\lambda_i = \Theta(\sqrt{n})$ for $i = 2, \dots, n$ and $a^2 + \sum_i a_i^2 = 1$ we have:

$$\mathbf{e}^\top G \mathbf{e} \leq a^2 \lambda + \Theta(\sqrt{n}).$$

Using the bound derived above of $\mathbf{e}^\top G \mathbf{e} \geq \mu n - o(1)$ and Lemma 15, we have:

$$\begin{aligned}\mu n - o(1) &\leq \lambda a_1^2 + \Theta(\sqrt{n}) \\ \frac{\mu n - \Theta(\sqrt{n})}{\mu n + \Theta(\sqrt{n})} &\leq a^2 \leq a\end{aligned}$$

from which we can conclude (with further manipulation):

$$1 - \frac{2\Theta(\sqrt{n})}{\mu n} = 1 - \frac{1}{\mu\Theta(\sqrt{n})} \leq a.$$

Consider now that a is the angle between \mathbf{e} and \mathbf{v} :

$$\frac{1}{\sqrt{n}} \sum_i v_i = \mathbf{e}^\top \mathbf{v} = a \geq 1 - \frac{1}{\mu\Theta(\sqrt{n})},$$

from which we obtain:

$$\sum_i v_i \geq \sqrt{n} - c,$$

for some constant c . \square

As a result so far, we have that

$$\begin{aligned}\lambda v_i &= (G\mathbf{v})_i = \mu \sum_i v_i + \sigma (S\mathbf{v})_i \\ &\geq \mu \sqrt{n} - C + \sigma \mathbf{g}^\top \mathbf{v}\end{aligned}$$

where $C = \mu c$ is a constant \mathbf{g} is some n -dimensional normally distributed i.i.d random vector. We would be done if we could show that the probability of the event $\mathbf{g}^\top \mathbf{v} > (1/\sigma)\mu\sqrt{n}$ occurs with probability $o(1)$, i.e., decays with the growth of n . The problem is that since \mathbf{g} stands for a row of S and because \mathbf{v} depends on S we cannot make the assumption that \mathbf{g} and \mathbf{v} are independent — thus a straightforward tail bound would not be appropriate. The remainder of the proof below was contributed by Ofer Zeitouni where care is taken to decouple the dependency between \mathbf{g} and \mathbf{v} .

Proof of Theorem 10: Let $D(c)$ be the set of vectors in R^n satisfying Lemma 16:

$$D(c) = \left\{ \mathbf{v} \in R^n : \|\mathbf{v}\| = 1, \sum_i v_i \geq \sqrt{n} - c \right\},$$

and let $\mathbf{g} \in R^n$ be a vector of i.i.d. standard Normal distribution $N(0, 1)$. We would like to analyze the probability of the event

$$F(g) = \left\{ \exists \mathbf{v} \in D(c) \text{ s.t. } \mathbf{g}^\top \mathbf{v} \geq \frac{\mu}{\sigma} \sqrt{n} \right\} \quad \mathbf{g} \in R^n, \text{ in the case where } g_i \sim N(0, 1).$$

In particular we would like to show that the probability $P_{g_i \sim N(0,1)}(F(g))$ belongs to $o(1)$, i.e., decays with the growth of n .

Let $\mathbf{v} = \mathbf{e} + \mathbf{f}$ where $\mathbf{e} = \frac{1}{\sqrt{n}}\mathbf{1}$ was defined above and \mathbf{f} is the residual. From the constraint $\|\mathbf{v}\|^2 = 1$ we obtain a constraint on \mathbf{f} :

$$\frac{2}{\sqrt{n}} \sum_i f_i + \sum_i f_i^2 = 0 \tag{10}$$

Given that $\mathbf{v} \in D(c)$ we obtain:

$$\sum_i v_i = \sqrt{n} \mathbf{v}^\top \mathbf{e} = \sqrt{n} + \sum_i f_i \geq \sqrt{n} - c,$$

from which obtain another constraint on \mathbf{f} :

$$-\sum_i f_i \leq c \tag{11}$$

Combining both constraints (10) and (11) we arrive at:

$$\|\mathbf{f}\|^2 \leq \frac{2c}{\sqrt{n}} \tag{12}$$

The expression $\mathbf{g}^\top \mathbf{v}$ can be broken down to a sum of two terms $\mathbf{g}^\top \mathbf{e}$ and $\mathbf{g}^\top \mathbf{f}$. The first of these two terms is $o(1)$ by the law of large numbers, and so:

$$\begin{aligned} \mathbf{g}^\top \mathbf{v} &= \mathbf{g}^\top \mathbf{e} + \mathbf{g}^\top \mathbf{f} \leq o(1) + \|\mathbf{g}\| \|\mathbf{f}\| \\ &\leq o(1) + \|\mathbf{g}\| \left(\frac{\sqrt{2c}}{n^{1/4}} \right) \end{aligned}$$

$\|\mathbf{g}\|$ distributes according to the χ distribution with n degrees of freedom, which concentrates around \sqrt{n} . Therefore, with probability $1 - o(1)$, $\|\mathbf{g}\| = \Theta(\sqrt{n})$. The probability that $\mathbf{g}^\top \mathbf{v} \geq \Theta(\sqrt{n})$ is proportional to the probability that $\|\mathbf{g}\| \geq n^{3/4}$, which by the Gaussian tail bound decays exponentially

with the growth of n . Since the probability that each entry of v is negative decays exponentially, i.e., $p(v_i < 0) < e^{-Cn}$, for some constant C , then by the union-bound the union of such events $p(v_1 < 0 \cup \dots \cup v_n < 0)$ is bounded from above by ne^{-Cn} which decays exponentially with the growth of n . \square

Lemma 12 Let \mathbf{g} be a random n -vector of i.i.d bounded variables, i.e., for each $i = 1..n$, $|\mathbf{g}_i| \leq M$. The following holds for some constant C :

$$P(\|\mathbf{g}\|^2 \geq Dn^{1/2+\epsilon}) \leq \exp\left(-\frac{C^2 D^2 n^{2\epsilon}}{M^2}\right)$$

Proof: We will apply Hoeffding's inequality to the random variable $\frac{1}{n}\|\mathbf{g}\|^2$, which has a mean μ that does not depend on n .

Assume $\gamma \geq Dn^{-1/2+\epsilon}$, where $\epsilon > 1/2$. For some $n > \hat{n}$, and for some c , $\gamma - \mu \geq c\gamma$. We get:

$$P\left(\frac{\|\mathbf{g}\|^2}{n} \geq \gamma\right) = P\left(\frac{\|\mathbf{g}\|^2}{n} - \mu \geq \gamma - \mu\right) \leq P\left(\frac{\|\mathbf{g}\|^2}{n} - \mu \geq c\gamma\right) .$$

Now, we can apply Hoeffding's one sided inequality and get:

$$P\left(\frac{1}{n}\|\mathbf{g}\|^2 - \mu \geq c\gamma\right) \leq \exp\left(-\frac{c^2 n \gamma^2}{M^2}\right) \leq \exp\left(-\frac{c^2 D^2 n^{2\epsilon}}{M^2}\right) .$$

\square

References

- H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *AI*, 69(1-2):279–305, 1991.
- A. Ben-Dor, N. Friedman, and Z. Yakhini. Class discovery in gene expression data. In *RECOMB*, 2001.
- A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *AI*, 97(1-2):245–271, 1997.
- O. Bousquet and D. J. L. Herrmann. On the complexity of learning the kernel matrix. In *NIPS*, 2003.
- P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, 1998.
- M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In *Ninth Int. Workshop on AI and Statistics*, 2003.
- O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.
- F. R. K. Chung. *Spectral Graph Theory*. AMS, 1998.

- Z. Furedi and J. Komlos. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3): 233–241, 1981.
- L. E. Gibbons, D. W. Hearn, P. M. Pardalos, and M. V. Ramana. Continuous characterizations of the maximum clique problem. *Math. Oper. Res.*, 22:754–768, 1997.
- G. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1996.
- T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
- I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research, Special issue on special feature*, 3:389–422, 2003.
- I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.
- K. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, 1970.
- K. Hall. ‘gene shaving’ as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 1(2):1–21, 2000.
- W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
- K. Kira and L. Rendell. A practical approach to feature selection. In *Ninth Int. Workshop on Machine Learning*, 1992.
- J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Journal of Information and Computation*, 132(1):1–63, 1997.
- R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.
- P. Langley and W. Iba. Average-case analysis of a nearest neighbor algorithm. In *13th Int. Joint Conf. on Artificial Intelligence*, 1993.
- D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- D. D. Lewis. Feature selection and feature extraction for text categorization. In *Speech and Natural Language Workshop*, 1992.
- R. Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988. ISSN 0018-9162. doi: <http://dx.doi.org/10.1109/2.36>.
- P. M. Long and V. B. Vega. Boosting and microarray data. *Machine Learning*, 52:31–44, 2003.

- M. L. Mehta. *Random Matrices*. Academic Press, 1991.
- B. Mohar. The laplacian spectrum of graphs. In Y. Alavi et al., editor, *Graph Theory, Combinatorics and Applications*. Wiley, New York, 1991.
- T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem by turan. *Canadian Journal of Math.*, 17:533–540, 1965.
- A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.
- B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(13):607–609, 1996.
- M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.
- P. Perona and W. T. Freeman. A factorization approach to grouping. In *European Conference of Computer Vision (ECCV)*, 1998.
- S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. H. Kim, L. C. Goumnerova, P.McL. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, and T. R. Golub. Gene expression-based classification and outcome prediction of central nervous system embryonal tumors. *Nature*, 415(24):436–442, 2002.
- S. Ramaswamy. Personal communication.
- R. Rifkin, G. Yeo, and T. Poggio. *Regularized Least Squares Classification*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*. IOS Press, Amsterdam, 2003.
- S. Sarkar and K. L. Boyer. Quantitative measures of change based on feature organization: eigenvalues and eigenvectors. *CVIU*, 71(1):110–136, 1998.
- B. Scholkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- A. Shashua and L. Wolf. Kernel feature selection with side data using a spectral approach. In T. Pajdla and J. Matas, editors, *ECCV (3)*, volume 3023 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2004. ISBN 3-540-21982-X.
- J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.
- M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J.L Kutok, R.C Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, T. S. Ray, M.A Koval, K.W Last, A. Norton, T. A. Lister, J. Mesirov, D. S. Neuberg, E. S. Lander, J. C. Aster, and T. R. Golub. Diffuse large b-cell lymphoma outcome prediction by gene expression profiling and supervised machine learning. *Nature Medicine*, 8(1): 68–74, 2002.

- N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.
- L. J. van 't Veer, H. Dai, M. J. van de Vijver, Y. D. He, A.A Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, A. T. Witteveen, G. J. Schreiber, R. M. Kerkhoven, C. Roberts, P. S. Linsley, R. Bernards, and S. H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, 2002.
- V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1998.
- N. Vasconcelos. Feature selection by maximum marginal diversity: optimality and implications for visual recognition. In *CVPR (1)*, pages 762–772. IEEE Computer Society, 2003. ISBN 0-7695-1900-8.
- P. Viola and M. Jones. Robust real-time object detection. Technical Report CRL-2001-1, Compaq Cambridge Research Lab, 2001.
- Y. Weiss. Segmentation using eigenvectors: A unifying view. In *ICCV*, 1999.
- J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. *NIPS*, 2001.
- E. P. Wigner. On the distribution of the roots of certain symmetric matrices. *Ann. of Math.(2)*, 67: 325–327, 1958.
- L. Wolf and A. Shashua. Kernel principal angles for classification machines with applications to image sequence interpretation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.
- L. Wolf, A. Shashua, and S. Mukherjee. Gene selection via a spectral approach. In *post CVPR IEEE Workshop on Computer Vision Methods for Bioinformatics (CVMB)*, 2005.

Working Set Selection Using Second Order Information for Training Support Vector Machines

Rong-En Fan
Pai-Hsuen Chen
Chih-Jen Lin

B90098@CSIE.NTU.EDU.TW
 R90008@CSIE.NTU.EDU.TW
 CJLIN@CSIE.NTU.EDU.TW

*Department of Computer Science, National Taiwan University
 Taipei 106, Taiwan*

Editor: Thorsten Joachims

Abstract

Working set selection is an important step in decomposition methods for training support vector machines (SVMs). This paper develops a new technique for working set selection in SMO-type decomposition methods. It uses second order information to achieve fast convergence. Theoretical properties such as linear convergence are established. Experiments demonstrate that the proposed method is faster than existing selection methods using first order information.

Keywords: support vector machines, decomposition methods, sequential minimal optimization, working set selection

1. Introduction

Support vector machines (SVMs) (Boser et al., 1992; Cortes and Vapnik, 1995) are a useful classification method. Given instances $\mathbf{x}_i, i = 1, \dots, l$ with labels $y_i \in \{1, -1\}$, the main task in training SVMs is to solve the following quadratic optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, i = 1, \dots, l, \\ & \mathbf{y}^T \boldsymbol{\alpha} = 0, \end{aligned} \tag{1}$$

where \mathbf{e} is the vector of all ones, C is the upper bound of all variables, Q is an l by l symmetric matrix with $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function.

The matrix Q is usually fully dense and may be too large to be stored. Decomposition methods are designed to handle such difficulties (e.g., Osuna et al., 1997; Joachims, 1998; Platt, 1998; Chang and Lin, 2001). Unlike most optimization methods which update the whole vector $\boldsymbol{\alpha}$ in each step of an iterative process, the decomposition method modifies only a subset of $\boldsymbol{\alpha}$ per iteration. This subset, denoted as the working set B , leads to a small sub-problem to be minimized in each iteration. An extreme case is the Sequential Minimal Optimization (SMO) (Platt, 1998), which restricts B to have only two elements. Then in each iteration one does not require any optimization software in order to solve a simple two-variable problem. This method is sketched in the following:

Algorithm 1 (SMO-type decomposition method)

1. Find α^1 as the initial feasible solution. Set $k = 1$.
2. If α^k is an optimal solution of (1), stop. Otherwise, find a *two-element* working set $B = \{i, j\} \subset \{1, \dots, l\}$. Define $N \equiv \{1, \dots, l\} \setminus B$ and α_B^k and α_N^k to be sub-vectors of α^k corresponding to B and N , respectively.
3. Solve the following sub-problem with the variable α_B :

$$\begin{aligned}
 \min_{\alpha_B} \quad & \frac{1}{2} [\alpha_B^T \quad (\alpha_N^k)^T] \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} - [\mathbf{e}_B^T \quad \mathbf{e}_N^T] \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} \\
 & = \frac{1}{2} \alpha_B^T Q_{BB} \alpha_B + (-\mathbf{e}_B + Q_{BN} \alpha_N^k)^T \alpha_B + \text{constant} \\
 & = \frac{1}{2} [\alpha_i \quad \alpha_j] \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + (-\mathbf{e}_B + Q_{BN} \alpha_N^k)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \text{constant} \\
 \text{subject to} \quad & 0 \leq \alpha_i, \alpha_j \leq C, \\
 & y_i \alpha_i + y_j \alpha_j = -\mathbf{y}_N^T \alpha_N^k,
 \end{aligned} \tag{2}$$

where $\begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$ is a permutation of the matrix Q .

4. Set α_B^{k+1} to be the optimal solution of (2) and $\alpha_N^{k+1} \equiv \alpha_N^k$. Set $k \leftarrow k + 1$ and goto Step 2.

Note that the set B changes from one iteration to another, but to simplify the notation, we just use B instead of B^k .

Since only few components are updated per iteration, for difficult problems, the decomposition method suffers from slow convergences. Better methods of working set selection could reduce the number of iterations and hence are an important research issue. Existing methods mainly rely on the violation of the optimality condition, which also corresponds to first order (i.e., gradient) information of the objective function. Past optimization research indicates that using second order information generally leads to faster convergence. Now (1) is a quadratic programming problem, so second order information directly relates to the decrease of the objective function. There are several attempts (e.g., Lai et al., 2003a,b) to find working sets based on the reduction of the objective value, but these selection methods are only heuristics without convergence proofs. Moreover, as such techniques cost more than existing ones, fewer iterations may not lead to shorter training time. This paper develops a simple working set selection using second order information. It can be extended for indefinite kernel matrices. Experiments demonstrate that the training time is shorter than existing implementations.

This paper is organized as follows. In Section 2, we discuss existing methods of working set selection and propose a new strategy. Theoretical properties of using the new selection technique are in Section 3. In Section 4 we extend the proposed selection method to other SVM formulas such as ν -SVM. A detailed experiment is in Section 5. We then in Section 6 discuss and compare some variants of the proposed selection method. Finally, Section 7 concludes this research work. A pseudo code of the proposed method is in the Appendix.

2. Existing and New Working Set Selections

In this section, we discuss existing methods of working set selection and then propose a new approach.

2.1 Existing Selections

Currently a popular way to select the working set B is via the “maximal violating pair:”

WSS 1 (Working set selection via the “maximal violating pair”)

1. Select

$$\begin{aligned} i &\in \arg \max_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{up}}(\boldsymbol{\alpha}^k)\}, \\ j &\in \arg \min_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k)\}, \end{aligned}$$

where

$$\begin{aligned} I_{\text{up}}(\boldsymbol{\alpha}) &\equiv \{t \mid \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}, \text{ and} \\ I_{\text{low}}(\boldsymbol{\alpha}) &\equiv \{t \mid \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}. \end{aligned} \tag{3}$$

2. Return $B = \{i, j\}$.

This working set was first proposed in Keerthi et al. (2001) and is used in, for example, the software LIBSVM (Chang and Lin, 2001). WSS 1 can be derived through the Karush-Kuhn-Tucker (KKT) optimality condition of (1): A vector $\boldsymbol{\alpha}$ is a stationary point of (1) if and only if there is a number b and two nonnegative vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ such that

$$\begin{aligned} \nabla f(\boldsymbol{\alpha}) + b\mathbf{y} &= \boldsymbol{\lambda} - \boldsymbol{\mu}, \\ \lambda_i \alpha_i &= 0, \mu_i (C - \alpha_i) = 0, \lambda_i \geq 0, \mu_i \geq 0, i = 1, \dots, l, \end{aligned}$$

where $\nabla f(\boldsymbol{\alpha}) \equiv Q\boldsymbol{\alpha} - \mathbf{e}$ is the gradient of $f(\boldsymbol{\alpha})$. This condition can be rewritten as

$$\nabla f(\boldsymbol{\alpha})_i + by_i \geq 0 \quad \text{if } \alpha_i < C, \tag{4}$$

$$\nabla f(\boldsymbol{\alpha})_i + by_i \leq 0 \quad \text{if } \alpha_i > 0. \tag{5}$$

Since $y_i = \pm 1$, by defining $I_{\text{up}}(\boldsymbol{\alpha})$ and $I_{\text{low}}(\boldsymbol{\alpha})$ as in (3), and rewriting (4)-(5) to

$$\begin{aligned} -y_i \nabla f(\boldsymbol{\alpha})_i &\leq b, \forall i \in I_{\text{up}}(\boldsymbol{\alpha}), \text{ and} \\ -y_i \nabla f(\boldsymbol{\alpha})_i &\geq b, \forall i \in I_{\text{low}}(\boldsymbol{\alpha}), \end{aligned}$$

a feasible $\boldsymbol{\alpha}$ is a stationary point of (1) if and only if

$$m(\boldsymbol{\alpha}) \leq M(\boldsymbol{\alpha}), \tag{6}$$

where

$$m(\boldsymbol{\alpha}) \equiv \max_{i \in I_{\text{up}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i, \text{ and } M(\boldsymbol{\alpha}) \equiv \min_{i \in I_{\text{low}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i.$$

Note that $m(\boldsymbol{\alpha})$ and $M(\boldsymbol{\alpha})$ are well defined except a rare situation where all $y_i = 1$ (or -1). In this case the zero vector is the only feasible solution of (1), so the decomposition method stops at the first iteration.

Following Keerthi et al. (2001), we define a “violating pair” of the condition (6).

Definition 1 (Violating pair) *If $i \in I_{\text{up}}(\boldsymbol{\alpha}), j \in I_{\text{low}}(\boldsymbol{\alpha})$, and $-y_i \nabla f(\boldsymbol{\alpha})_i > -y_j \nabla f(\boldsymbol{\alpha})_j$, then $\{i, j\}$ is a “violating pair.”*

From (6), indices $\{i, j\}$ which most violate the optimality condition are a natural choice of the working set. They are called a “maximal violating pair” in WSS 1. It is known that violating pairs are important in the working set selection:

Theorem 2 (Hush and Scovel, 2003) *Assume Q is positive semi-definite. SMO-type methods have the strict decrease of the function value (i.e., $f(\boldsymbol{\alpha}^{k+1}) < f(\boldsymbol{\alpha}^k), \forall k$) if and only if B is a violating pair.*

Interestingly, the maximal violating pair is related to first order approximation of $f(\boldsymbol{\alpha})$. As explained below, $\{i, j\}$ selected via WSS 1 satisfies

$$\{i, j\} = \arg \min_{B: |B|=2} \text{Sub}(B), \quad (7)$$

where

$$\text{Sub}(B) \equiv \min_{\mathbf{d}_B} \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B \quad (8a)$$

$$\text{subject to } \mathbf{y}_B^T \mathbf{d}_B = 0, \quad (8b)$$

$$d_t \geq 0, \text{ if } \alpha_t^k = 0, t \in B, \quad (8b)$$

$$d_t \leq 0, \text{ if } \alpha_t^k = C, t \in B, \quad (8c)$$

$$-1 \leq d_t \leq 1, t \in B. \quad (8d)$$

Problem (7) was first considered in Joachims (1998). By defining $\mathbf{d}^T \equiv [\mathbf{d}_B^T, \mathbf{0}_N^T]$, the objective function (8a) comes from minimizing *first order approximation* of $f(\boldsymbol{\alpha}^k + \mathbf{d})$:

$$\begin{aligned} f(\boldsymbol{\alpha}^k + \mathbf{d}) &\approx f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)^T \mathbf{d} \\ &= f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B. \end{aligned}$$

The constraint $\mathbf{y}_B^T \mathbf{d}_B = 0$ is from $\mathbf{y}^T(\boldsymbol{\alpha}^k + \mathbf{d}) = 0$ and $\mathbf{y}^T \boldsymbol{\alpha}^k = 0$. The condition $0 \leq \alpha_t \leq C$ leads to inequalities (8b) and (8c). As (8a) is a linear function, the inequalities $-1 \leq d_t \leq 1, t \in B$ avoid that the objective value goes to $-\infty$.

A first look at (7) indicates that we may have to check all $\binom{l}{2}$ B 's in order to find an optimal set. Instead, WSS 1 efficiently solves (7) in $O(l)$ steps. This result is discussed in Lin (2001b, Section II), where more general settings ($|B|$ is any even integer) are considered. The proof for $|B| = 2$ is easy, so we give it in Appendix A for completeness.

The convergence of the decomposition method using WSS 1 is proved in Lin (2001b, 2002).

2.2 A New Working Set Selection

Instead of using first order approximation, we may consider more accurate second order information. As f is a quadratic,

$$\begin{aligned} f(\boldsymbol{\alpha}^k + \mathbf{d}) - f(\boldsymbol{\alpha}^k) &= \nabla f(\boldsymbol{\alpha}^k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 f(\boldsymbol{\alpha}^k) \mathbf{d} \\ &= \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B + \frac{1}{2} \mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB} \mathbf{d}_B \end{aligned} \quad (9)$$

is exactly the reduction of the objective value. Thus, by replacing the objective function of (8) with (9), a selection method using *second order information* is

$$\min_{B:|B|=2} \text{Sub}(B), \quad (10)$$

where

$$\text{Sub}(B) \equiv \min_{\mathbf{d}_B} \frac{1}{2} \mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB} \mathbf{d}_B + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B \quad (11a)$$

$$\text{subject to } \mathbf{y}_B^T \mathbf{d}_B = 0, \quad (11b)$$

$$d_t \geq 0, \text{ if } \alpha_t^k = 0, t \in B, \quad (11c)$$

$$d_t \leq 0, \text{ if } \alpha_t^k = C, t \in B. \quad (11d)$$

Note that inequality constraints $-1 \leq d_t \leq 1, t \in B$ in (8) are removed, as later we will see that the optimal value of (11) does not go to $-\infty$. Though one expects (11) is better than (8), $\min_{B:|B|=2} \text{Sub}(B)$ in (10) becomes a challenging task. Unlike (7)-(8), which can be efficiently solved by WSS 1, for (10) and (11) there is no available way to avoid checking all $\binom{l}{2}$ B 's. Note that except the working set selection, the main task per decomposition iteration is on calculating the two kernel columns Q_{ti} and Q_{tj} , $t = 1, \dots, l$. This requires $O(l)$ operations and is needed only if Q is not stored. Therefore, each iteration can become l times more expensive if an $O(l^2)$ working set selection is used. Moreover, from simple experiments we know that the number of iterations is, however, not decreased l times. Therefore, an $O(l^2)$ working set selection is impractical.

A viable implementation of using second order information is thus to heuristically check several B 's only. We propose the following new selection:

WSS 2 (Working set selection using second order information)

1. Select

$$i \in \arg \max_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{up}}(\boldsymbol{\alpha}^k)\}.$$

2. Consider $\text{Sub}(B)$ defined in (11) and select

$$j \in \arg \min_t \{\text{Sub}(\{i, t\}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i\}. \quad (12)$$

3. Return $B = \{i, j\}$.

By using the same i as in WSS 1, we check only $O(l)$ possible B 's to decide j . Alternatively, one may choose $j \in \arg M(\boldsymbol{\alpha}^k)$ and search for i by a way similar to (12)¹. In fact, such a selection is the same as swapping labels \mathbf{y} first and then applying WSS 2, so the performance should not differ much. It is certainly possible to consider other heuristics, and the main concern is how good they are if compared to the one by fully checking all $\binom{l}{2}$ sets. In Section 7 we will address this issue. Experiments indicate that a full check does not reduce iterations of using WSS 2 much. Thus WSS 2 is already a very good way of using second order information.

1. To simplify the notations, we denote $\arg M(\boldsymbol{\alpha})$ as $\arg \min_{t \in I_{\text{low}}(\boldsymbol{\alpha})} -y_t \nabla f(\boldsymbol{\alpha})_t$ and $\arg m(\boldsymbol{\alpha})$ as $\arg \max_{t \in I_{\text{up}}(\boldsymbol{\alpha})} -y_t \nabla f(\boldsymbol{\alpha})_t$, respectively.

Despite the above issue of how to effectively use second order information, the real challenge is whether the new WSS 2 can cause shorter training time than WSS 1. Now the two selection methods differ only in selecting j , so we can also consider WSS 2 as a direct attempt to improve WSS 1. The following theorem shows that one could efficiently solve (11), so the working set selection WSS 2 does not cost a lot more than WSS 1.

Theorem 3 *If $B = \{i, j\}$ is a violating pair and $K_{ii} + K_{jj} - 2K_{ij} > 0$, then (11) has the optimal objective value*

$$-\frac{(-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j)^2}{2(K_{ii} + K_{jj} - 2K_{ij})}.$$

Proof Define $\hat{d}_i \equiv y_i d_i$ and $\hat{d}_j \equiv y_j d_j$. From $\mathbf{y}_B^T \mathbf{d}_B = 0$, we have $\hat{d}_i = -\hat{d}_j$ and

$$\begin{aligned} & \frac{1}{2} \begin{bmatrix} d_i & d_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} + \begin{bmatrix} \nabla f(\boldsymbol{\alpha}^k)_i & \nabla f(\boldsymbol{\alpha}^k)_j \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} \\ &= \frac{1}{2} (K_{ii} + K_{jj} - 2K_{ij}) \hat{d}_j^2 + (-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j) \hat{d}_j. \end{aligned} \quad (13)$$

Since $K_{ii} + K_{jj} - 2K_{ij} > 0$ and B is a violating pair, we can define

$$a_{ij} \equiv K_{ii} + K_{jj} - 2K_{ij} > 0 \quad \text{and} \quad b_{ij} \equiv -y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j > 0. \quad (14)$$

Then (13) has the minimum at

$$\hat{d}_j = -\hat{d}_i = -\frac{b_{ij}}{a_{ij}} < 0, \quad (15)$$

and

$$\text{the objective function (11a)} = -\frac{b_{ij}^2}{2a_{ij}}.$$

Moreover, we can show that \hat{d}_i and \hat{d}_j (d_i and d_j) indeed satisfy (11c)-(11d). If $j \in I_{\text{low}}(\boldsymbol{\alpha}^k)$, $\alpha_j^k = 0$ implies $y_j = -1$ and hence $d_j = y_j \hat{d}_j > 0$, a condition required by (11c). Other cases are similar. Thus \hat{d}_i and \hat{d}_j defined in (15) are optimal for (11). ■

Note that if K is positive definite, then for any $i \neq j$, $K_{ii} + K_{jj} - 2K_{ij} > 0$. Using Theorem 3, (12) in WSS 2 is reduced to a very simple form:

$$j \in \arg \min_t \left\{ -\frac{b_{it}^2}{a_{it}} \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i \right\},$$

where a_{it} and b_{it} are defined in (14). If K is not positive definite, the leading coefficient a_{ij} of (13) may be non-positive. This situation will be addressed in the next sub-section.

Note that (8) and (11) are used only for selecting the working set, so they do not have to maintain the feasibility $0 \leq \alpha_i^k + d_i \leq C, \forall i \in B$. On the contrary, feasibility must hold for the sub-problem (2) used to obtain $\boldsymbol{\alpha}^{k+1}$ after B is determined. There are some earlier attempts to use second order information for selecting working sets (e.g., Lai et al., 2003a,b), but they always check the feasibility. Then solving sub-problems during the

selection procedure is more complicated than solving the sub-problem (11). Besides, these earlier approaches do not provide any convergence analysis. In Section 6 we will investigate the issue of maintaining feasibility in the selection procedure, and explain why using (11) is better.

2.3 Non-Positive Definite Kernel Matrices

Theorem 3 does not hold if $K_{ii} + K_{jj} - 2K_{ij} \leq 0$. For the linear kernel, sometimes K is only positive semi-definite, so it is possible that $K_{ii} + K_{jj} - 2K_{ij} = 0$. Moreover, some existing kernel functions (e.g., sigmoid kernel) are not the inner product of two vectors, so K is even not positive semi-definite. Then $K_{ii} + K_{jj} - 2K_{ij} < 0$ may occur and (13) is a concave objective function.

Once B is decided, the same difficulty occurs for the sub-problem (2) to obtain $\boldsymbol{\alpha}^{k+1}$. Note that (2) differs from (11) only in constraints; (2) strictly requires the feasibility $0 \leq \alpha_i + d_i \leq C, \forall t \in B$. Therefore, (2) also has a concave objective function if $K_{ii} + K_{jj} - 2K_{ij} < 0$. In this situation, (2) may possess multiple local minima. Moreover, there are difficulties in proving the convergence of the decomposition methods (Palagi and Sciandrone, 2005; Chen et al., 2006). Thus, Chen et al. (2006) proposed adding an additional term to (2)'s objective function if $a_{ij} \equiv K_{ii} + K_{jj} - 2K_{ij} \leq 0$:

$$\begin{aligned} \min_{\alpha_i, \alpha_j} \quad & \frac{1}{2} [\alpha_i \quad \alpha_j] \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + (-\mathbf{e}_B + Q_{BN} \boldsymbol{\alpha}_N^k)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \\ & \frac{\tau - a_{ij}}{4} ((\alpha_i - \alpha_i^k)^2 + (\alpha_j - \alpha_j^k)^2) \\ \text{subject to} \quad & 0 \leq \alpha_i, \alpha_j \leq C, \\ & y_i \alpha_i + y_j \alpha_j = -\mathbf{y}_N^T \boldsymbol{\alpha}_N^k, \end{aligned} \quad (16)$$

where τ is a small positive number. By defining $\hat{d}_i \equiv y_i(\alpha_i - \alpha_i^k)$ and $\hat{d}_j \equiv y_j(\alpha_j - \alpha_j^k)$, (16)'s objective function, in a form similar to (13), is

$$\frac{1}{2} \tau \hat{d}_j^2 + b_{ij} \hat{d}_j, \quad (17)$$

where b_{ij} is defined as in (14). The new objective function is thus strictly convex. If $\{i, j\}$ is a violating pair, then a careful check shows that there is $\hat{d}_j < 0$ which leads to a negative value in (17) and maintains the feasibility of (16). Therefore, we can find $\boldsymbol{\alpha}^{k+1} \neq \boldsymbol{\alpha}^k$ satisfying $f(\boldsymbol{\alpha}^{k+1}) < f(\boldsymbol{\alpha}^k)$. More details are in Chen et al. (2006).

For selecting the working set, we consider a similar modification: If $B = \{i, j\}$ and a_{ij} is defined as in (14), then (11) is modified to:

$$\begin{aligned} \text{Sub}(B) \equiv \min_{d_B} \quad & \frac{1}{2} \mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB} \mathbf{d}_B + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B + \frac{\tau - a_{ij}}{4} (d_i^2 + d_j^2) \\ \text{subject to} \quad & \text{constraints of (11)}. \end{aligned} \quad (18)$$

Note that (18) differs from (16) only in constraints. In (18) we do not maintain the feasibility of $\alpha_t^k + d_t, t \in B$. We are allowed to do so because (18) is used only for identifying the working set B .

By reformulating (18) to (17) and following the same argument in Theorem 3, the optimal objective value of (18) is

$$-\frac{b_{ij}^2}{2\tau}.$$

Therefore, a generalized working set selection is as the following:

WSS 3

(Working set selection using second order information: any symmetric K)

1. Define a_{ts} and b_{ts} as in (14), and

$$\bar{a}_{ts} \equiv \begin{cases} a_{ts} & \text{if } a_{ts} > 0, \\ \tau & \text{otherwise.} \end{cases} \quad (19)$$

Select

$$\begin{aligned} i &\in \arg \max_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{up}}(\boldsymbol{\alpha}^k)\}, \\ j &\in \arg \min_t \left\{ -\frac{b_{it}^2}{\bar{a}_{it}} \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i \right\}. \end{aligned} \quad (20)$$

2. Return $B = \{i, j\}$.

In summary, an SMO-type decomposition method using WSS 3 for the working set selection is:

Algorithm 2 (An SMO-type decomposition method using WSS 3)

1. Find $\boldsymbol{\alpha}^1$ as the initial feasible solution. Set $k = 1$.
2. If $\boldsymbol{\alpha}^k$ is a stationary point of (1), stop. Otherwise, find a working set $B = \{i, j\}$ by WSS 3.
3. Let a_{ij} be defined as in (14). If $a_{ij} > 0$, solve the sub-problem (2). Otherwise, solve (16). Set $\boldsymbol{\alpha}_B^{k+1}$ to be the optimal point of the sub-problem.
4. Set $\boldsymbol{\alpha}_N^{k+1} \equiv \boldsymbol{\alpha}_N^k$. Set $k \leftarrow k + 1$ and goto Step 2.

In the next section we study theoretical properties of using WSS 3.

3. Theoretical Properties

To obtain theoretical properties of using WSS 3, we consider the work (Chen et al., 2006), which gives a general study of SMO-type decomposition methods. It considers Algorithm 2 but replaces WSS 3 with a general working set selection²:

WSS 4 (A general working set selection discussed in Chen et al., 2006)

1. Consider a fixed $0 < \sigma \leq 1$ for all iterations.

2. In fact, Chen et al. (2006) consider an even more general framework for selecting working sets, but for easy description, we discuss WSS 4 here.

2. Select any $i \in I_{\text{up}}(\boldsymbol{\alpha}^k), j \in I_{\text{low}}(\boldsymbol{\alpha}^k)$ satisfying

$$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \geq \sigma(m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)) > 0. \quad (21)$$

3. Return $B = \{i, j\}$.

Clearly (21) ensures the quality of the selected pair by linking it to the maximal violating pair. It is easy to see that WSS 3 is a special case of WSS 4: Assume $B = \{i, j\}$ is the set returned from WSS 3 and $\bar{j} \in \arg M(\boldsymbol{\alpha}^k)$. Since WSS 3 selects $i \in \arg m(\boldsymbol{\alpha}^k)$, with $\bar{a}_{ij} > 0$ and $\bar{a}_{i\bar{j}} > 0$, (20) in WSS 3 implies

$$\frac{-(-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j)^2}{\bar{a}_{ij}} \leq \frac{-(m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k))^2}{\bar{a}_{i\bar{j}}}.$$

Thus,

$$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \geq \sqrt{\frac{\min_{t,s} \bar{a}_{t,s}}{\max_{t,s} \bar{a}_{t,s}}} (m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)),$$

an inequality satisfying (21) for $\sigma = \sqrt{\min_{t,s} \bar{a}_{t,s} / \max_{t,s} \bar{a}_{t,s}}$.

Therefore, all theoretical properties proved in Chen et al. (2006) hold here. They are listed below.

It is known that the decomposition method may not converge to an optimal solution if using improper methods of working set selection. We thus must prove that the proposed selection leads to the convergence.

Theorem 4 (Asymptotic convergence (Chen et al., 2006, Theorem 3 and Corollary 1))

Let $\{\boldsymbol{\alpha}^k\}$ be the infinite sequence generated by the SMO-type method Algorithm 2. Then any limit point of $\{\boldsymbol{\alpha}^k\}$ is a stationary point of (1). Moreover, if Q is positive definite, $\{\boldsymbol{\alpha}^k\}$ globally converges to the unique minimum of (1).

As the decomposition method only asymptotically approaches an optimum, in practice, it is terminated after satisfying a stopping condition. For example, we can pre-specify a small tolerance $\epsilon > 0$ and check if the maximal violation is small enough:

$$m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k) \leq \epsilon. \quad (22)$$

Alternatively, one may check if the selected working set $\{i, j\}$ satisfies

$$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \leq \epsilon, \quad (23)$$

because (21) implies $m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k) \leq \epsilon/\sigma$. These are reasonable stopping criteria due to their closeness to the optimality condition (6). To avoid an infinite loop, we must have that under any $\epsilon > 0$, Algorithm 2 stops in a finite number of iterations. The finite termination of using (22) or (23) as the stopping condition is implied by (26) of Theorem 5 stated below.

Shrinking and caching (Joachims, 1998) are two effective techniques to make the decomposition method faster. The former removes some bounded components during iterations, so smaller reduced problems are considered. The latter allocates some memory space (called cache) to store recently used Q_{ij} , and may significantly reduce the number of kernel evaluations. The following theorem explains why these two techniques are useful in practice:

Theorem 5 (Finite termination and explanation of caching and shrinking techniques (Chen et al., 2006, Theorems 4 and 6))

Assume Q is positive semi-definite.

1. The following set is independent of any optimal solution $\bar{\alpha}$:

$$I \equiv \{i \mid -y_i \nabla f(\bar{\alpha})_i > M(\bar{\alpha}) \text{ or } -y_i \nabla f(\bar{\alpha})_i < m(\bar{\alpha})\}. \quad (24)$$

Problem (1) has unique and bounded optimal solutions at $\alpha_i, i \in I$.

2. Assume Algorithm 2 generates an infinite sequence $\{\alpha^k\}$. There is \bar{k} such that after $k \geq \bar{k}$, every $\alpha_i^k, i \in I$ has reached the unique and bounded optimal solution. It remains the same in all subsequent iterations and $\forall k \geq \bar{k}$:

$$i \notin \{t \mid M(\alpha^k) \leq -y_t \nabla f(\alpha^k)_t \leq m(\alpha^k)\}. \quad (25)$$

3. If (1) has an optimal solution $\bar{\alpha}$ satisfying $m(\bar{\alpha}) < M(\bar{\alpha})$, then $\bar{\alpha}$ is the unique solution and Algorithm 2 reaches it in a finite number of iterations.
4. If $\{\alpha^k\}$ is an infinite sequence, then the following two limits exist and are equal:

$$\lim_{k \rightarrow \infty} m(\alpha^k) = \lim_{k \rightarrow \infty} M(\alpha^k) = m(\bar{\alpha}) = M(\bar{\alpha}), \quad (26)$$

where $\bar{\alpha}$ is any optimal solution.

Finally, the following theorem shows that Algorithm 2 is linearly convergent under some assumptions:

Theorem 6 (Linear convergence (Chen et al., 2006, Theorem 8))

Assume problem (1) satisfies

1. Q is positive definite. Therefore, (1) has a unique optimal solution $\bar{\alpha}$.
2. The nondegeneracy condition. That is, the optimal solution $\bar{\alpha}$ satisfies that

$$\nabla f(\bar{\alpha})_i + \bar{b}y_i = 0 \text{ if and only if } 0 < \bar{\alpha}_i < C, \quad (27)$$

where $\bar{b} = m(\bar{\alpha}) = M(\bar{\alpha})$ according to Theorem 5.

For the sequence $\{\alpha^k\}$ generated by Algorithm 2, there are $c < 1$ and \bar{k} such that for all $k \geq \bar{k}$,

$$f(\alpha^{k+1}) - f(\bar{\alpha}) \leq c(f(\alpha^k) - f(\bar{\alpha})).$$

This theorem indicates how fast the SMO-type method Algorithm 2 converges. For any fixed problem (1) and a given tolerance ϵ , there is \bar{k} such that within

$$\bar{k} + O(1/\epsilon)$$

iterations,

$$|f(\alpha^k) - f(\bar{\alpha})| \leq \epsilon.$$

Note that $O(1/\epsilon)$ iterations are necessary for decomposition methods according to the analysis in Lin (2001a)³. Hence the result of linear convergence here is already the best worst case analysis.

3. Lin (2001a) gave a three-variable example and explained that the SMO-type method using WSS 1 is linearly convergent. A careful check shows that the same result holds for any method of working set selection.

4. Extensions

The proposed WSS 3 can be directly used for training support vector regression (SVR) and one-class SVM because they solve problems similar to (1). More detailed discussion about applying WSS 4 (and hence WSS 3) to SVR and one-class SVM is in Chen et al. (2006, Section IV).

Another formula which needs special attention is ν -SVM (Schölkopf et al., 2000), which solves a problem with one more linear constraint:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2} \boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{y}^T \boldsymbol{\alpha} = 0, \\ & \mathbf{e}^T \boldsymbol{\alpha} = \nu, \\ & 0 \leq \alpha_i \leq 1/l, i = 1, \dots, l, \end{aligned} \quad (28)$$

where \mathbf{e} is the vector of all ones and $0 \leq \nu \leq 1$.

Similar to (6), $\boldsymbol{\alpha}$ is a stationary point of (28) if and only if it satisfies

$$m_p(\boldsymbol{\alpha}) \leq M_p(\boldsymbol{\alpha}) \text{ and } m_n(\boldsymbol{\alpha}) \leq M_n(\boldsymbol{\alpha}), \quad (29)$$

where

$$\begin{aligned} m_p(\boldsymbol{\alpha}) &\equiv \max_{i \in I_{\text{up}}(\boldsymbol{\alpha}), y_i=1} -y_i \nabla f(\boldsymbol{\alpha})_i, & M_p(\boldsymbol{\alpha}) &\equiv \min_{i \in I_{\text{low}}(\boldsymbol{\alpha}), y_i=1} -y_i \nabla f(\boldsymbol{\alpha})_i, \text{ and} \\ m_n(\boldsymbol{\alpha}) &\equiv \max_{i \in I_{\text{up}}(\boldsymbol{\alpha}), y_i=-1} -y_i \nabla f(\boldsymbol{\alpha})_i, & M_n(\boldsymbol{\alpha}) &\equiv \min_{i \in I_{\text{low}}(\boldsymbol{\alpha}), y_i=-1} -y_i \nabla f(\boldsymbol{\alpha})_i. \end{aligned}$$

A detailed derivation is in, for example, Chen et al. (2006, Section VI).

In an SMO-type method for ν -SVM the selected working set $B = \{i, j\}$ must satisfy $y_i = y_j$. Otherwise, if $y_i \neq y_j$, then the two linear equalities make the sub-problem have only one feasible point $\boldsymbol{\alpha}_B^k$. Therefore, to select the working set, one considers positive (i.e., $y_i = 1$) and negative (i.e., $y_i = -1$) instances separately. Existing implementations such as LIBSVM (Chang and Lin, 2001) check violating pairs in each part and select the one with the largest violation. This strategy is an extension of WSS 1. By a derivation similar to that in Section 2, the selection can also be from first or second order approximation of the objective function. Using $\text{Sub}(\{i, j\})$ defined in (11), WSS 2 in Section 2 is modified to

WSS 5 (Extending WSS 2 for ν -SVM)

1. Find

$$\begin{aligned} i_p &\in \arg m_p(\boldsymbol{\alpha}^k), \\ j_p &\in \arg \min_t \{\text{Sub}(\{i_p, t\}) \mid y_t = 1, \boldsymbol{\alpha}_t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_{i_p} \nabla f(\boldsymbol{\alpha}^k)_{i_p}\}. \end{aligned}$$

2. Find

$$\begin{aligned} i_n &\in \arg m_n(\boldsymbol{\alpha}^k), \\ j_n &\in \arg \min_t \{\text{Sub}(\{i_n, t\}) \mid y_t = -1, \boldsymbol{\alpha}_t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_{i_n} \nabla f(\boldsymbol{\alpha}^k)_{i_n}\}. \end{aligned}$$

Problem	#data	#feat.	Problem	#data	#feat.	Problem	#data	#feat.
image	1,300	18	breast-cancer	690	10	abalone*	1,000	8
splice	1,000	60	diabetes	768	8	cadata*	1,000	8
tree	700	18	fourclass	862	2	cpusmall*	1,000	12
a1a	1,605	119	german.numer	1,000	24	mg	1,385	6
australian	683	14	w1a	2,477	300	space_ga*	1,000	6

Table 1: Data statistics for small problems (left two columns: classification, right column: regression). *: subset of the original problem.

3. Check $\text{Sub}(\{i_p, j_p\})$ and $\text{Sub}(\{i_n, j_n\})$. Return the set with a smaller value.

By Theorem 3 in Section 2, it is easy to solve $\text{Sub}(\{i_p, t\})$ and $\text{Sub}(\{i_n, t\})$ in the above procedure.

5. Experiments

In this section we aim at comparing the proposed WSS 3 with WSS 1, which selects the maximal violating pair. As indicated in Section 2, they differ only in finding the second element j : WSS 1 checks first order approximation of the objective function, but WSS 3 uses second order information.

5.1 Data and Experimental Settings

First, some small data sets (around 1,000 samples) including ten binary classification and five regression problems are investigated under various settings. Secondly, observations are further confirmed by using four large (more than 30,000 instances) classification problems. Data statistics are in Tables 1 and 3.

Problems `german.numer` and `australian` are from the Statlog collection (Michie et al., 1994). We select `space_ga` and `cadata` from StatLib (<http://lib.stat.cmu.edu/datasets>). The data sets `image`, `diabetes`, `covtype`, `breast-cancer`, and `abalone` are from the UCI machine learning repository (Blake and Merz, 1998). Problems `a1a` and `a9a` are compiled in Platt (1998) from the UCI “adult” data set. Problems `w1a` and `w8a` are also from Platt (1998). The `tree` data set was originally used in Bailey et al. (1993). The problem `mg` is a Mackey-Glass time series. The data sets `cpusmall` and `splice` are from the Delve archive (<http://www.cs.toronto.edu/~delve>). Problem `fourclass` is from Ho and Kleinberg (1996) and we further transform it to a two-class set. The problem `IJCNN1` is from the first problem of IJCNN 2001 challenge (Prokhorov, 2001).

For most data sets each attribute is linearly scaled to $[-1, 1]$. We do not scale `a1a`, `a9a`, `w1a`, and `w8a` as they take two values 0 and 1. Another exception is `covtype`, in which 44 of 54 features have 0/1 values. We scale only the other ten features to $[0, 1]$. All data are available at <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/>. We use LIBSVM (version 2.71) (Chang and Lin, 2001), an implementation of WSS 1, for experiments. An easy modification to WSS 3 ensures that two codes differ only in the working set implementation. We set $\tau = 10^{-12}$ in WSS 3.

Different SVM parameters such as C in (1) and kernel parameters affect the training time. It is difficult to evaluate the two methods under every parameter setting. To have a fair comparison, we simulate how one uses SVM in practice and consider the following procedure:

1. “Parameter selection” step: Conduct five-fold cross validation to find the best one within a given set of parameters.
2. “Final training” step: Train the whole set with the best parameter to obtain the final model.

For each step we check time and iterations using the two methods of working set selection. For some extreme parameters (e.g., very large or small values) in the “parameter selection” step, the decomposition method converges very slowly, so the comparison shows if the proposed WSS 3 saves time under difficult situations. On the other hand, the best parameter usually locates in a more normal region, so the “final training” step tests if WSS 3 is competitive with WSS 1 for easier cases.

The behavior of using different kernels is a concern, so we thoroughly test four commonly used kernels:

1. RBF kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}.$$

2. Linear kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j.$$

3. Polynomial kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + 1)^d.$$

4. Sigmoid kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + d).$$

Note that this function cannot be represented as $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ under some parameters. Then the matrix Q is not positive semi-definite. Experimenting with this kernel tests if our extension to indefinite kernels in Section 2.3 works well or not.

Parameters used for each kernel are listed in Table 2. Note that as SVR has an additional parameter ϵ , to save the running time, for other parameters we may not consider as many values as in classification.

It is important to check how WSS 3 performs after incorporating shrinking and caching strategies. Such techniques may effectively save kernel evaluations at each iteration, so the higher cost of WSS 3 is a concern. We consider various settings:

1. With or without shrinking.
2. Different cache size: First a 40MB cache allows the whole kernel matrix to be stored in the computer memory. Second, we allocate only 100K, so cache misses may happen and more kernel evaluations are needed. The second setting simulates the training of large-scale sets whose kernel matrices cannot be stored.

Kernel	Problem type	$\log_2 C$	$\log_2 \gamma$	$\log_2 \epsilon$	d
RBF	Classification	-5, 15, 2	3, -15, -2		
	Regression	-1, 15, 2	3, -15, -2	-8, -1, 1	
Linear	Classification	-3, 5, 2			
	Regression	-3, 5, 2		-8, -1, 1	
Polynomial	Classification	-3, 5, 2	-5, -1, 1		2, 4, 1
	Regression	-3, 5, 2	-5, -1, 1	-8, -1, 1	2, 4, 1
Sigmoid	Classification	-3, 12, 3	-12, 3, 3		-2.4, 2.4, 0.6
	Regression	-3, 9, 3	$\gamma = \frac{1}{\#features}$	-8, -1, 3	-2.4, 2.4, 0.6

Table 2: Parameters used for various kernels: values of each parameter are from a uniform discretization of an interval. We list the left, right end points and the space for discretization. For example, $-5, 15, 2$ for $\log_2 C$ means $\log_2 C = -5, -3, \dots, 15$.

For each kernel, we give two figures showing results of “parameter selection” and “final training” steps, respectively. We further separate each figure to two scenarios: without/with shrinking, and present three ratios between using WSS 3 and using WSS 1:

$$\begin{aligned}
 \text{ratio 1} &\equiv \frac{\# \text{ iter. by Alg. 2 with WSS 3}}{\# \text{ iter. by Alg. 2 with WSS 1}}, \\
 \text{ratio 2} &\equiv \frac{\text{time by Alg. 2 (WSS 3, 100K cache)}}{\text{time by Alg. 2 (WSS 1, 100K cache)}}, \\
 \text{ratio 3} &\equiv \frac{\text{time by Alg. 2 (WSS 3, 40M cache)}}{\text{time by Alg. 2 (WSS 1, 40M cache)}}.
 \end{aligned}$$

Note that the number of iterations is independent of the cache size. For the “parameter selection” step, time (or iterations) of all parameters is summed up before calculating the ratio. In general the “final training” step is very fast, so the timing result may not be accurate. Hence we repeat this step several times to obtain more reliable timing values. Figures 1-8 present obtained ratios. They are in general smaller than one, so using WSS 3 is really better than using WSS 1. Before describing other results, we explain an interesting observation: In these figures, if shrinking is not used, in general

$$\text{ratio 1} \leq \text{ratio 2} \leq \text{ratio 3}. \tag{30}$$

Under the two very different cache sizes, one is too small to store the kernel matrix, but the other is large enough. Thus, roughly we have

$$\begin{aligned}
 \text{time per Alg. 2 iteration (100K cache)} &\approx \text{Calculating two } Q \text{ columns} + \text{Selection,} \\
 \text{time per Alg. 2 iteration (40M cache)} &\approx \text{Selection.}
 \end{aligned} \tag{31}$$

If shrinking is not used, the optimization problem is not reduced and hence

$$\frac{\text{time by Alg. 2}}{\# \text{ iter. of Alg. 2}} \approx \text{cost per iteration} \approx \text{constant}. \tag{32}$$

WORKING SET SELECTION FOR TRAINING SVMs

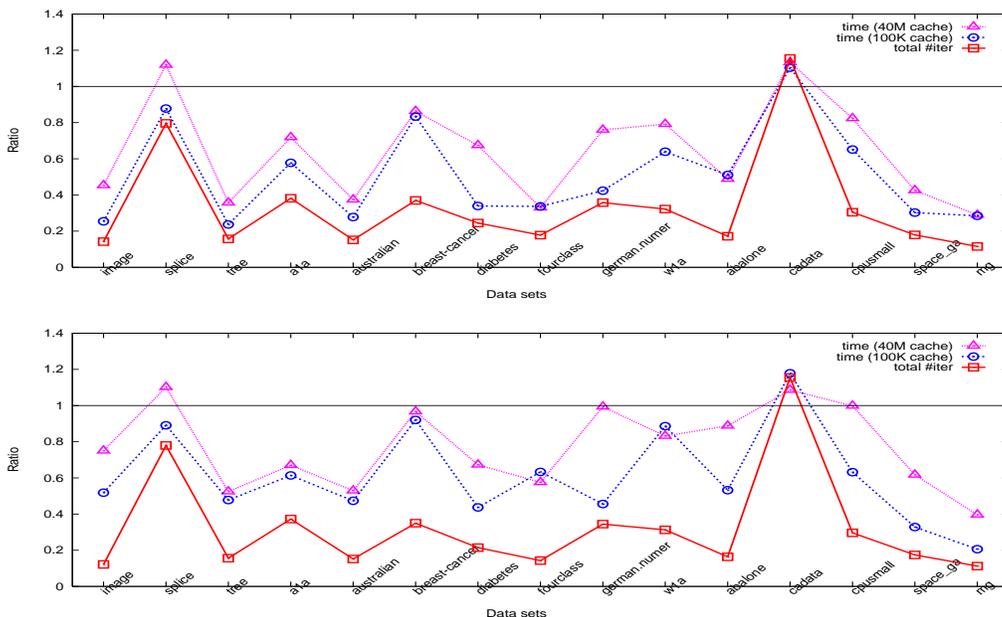


Figure 1: Iteration and time ratios between WSS 3 and 1 using the RBF kernel for the “parameter selection” step (top: without shrinking, bottom: with shrinking).

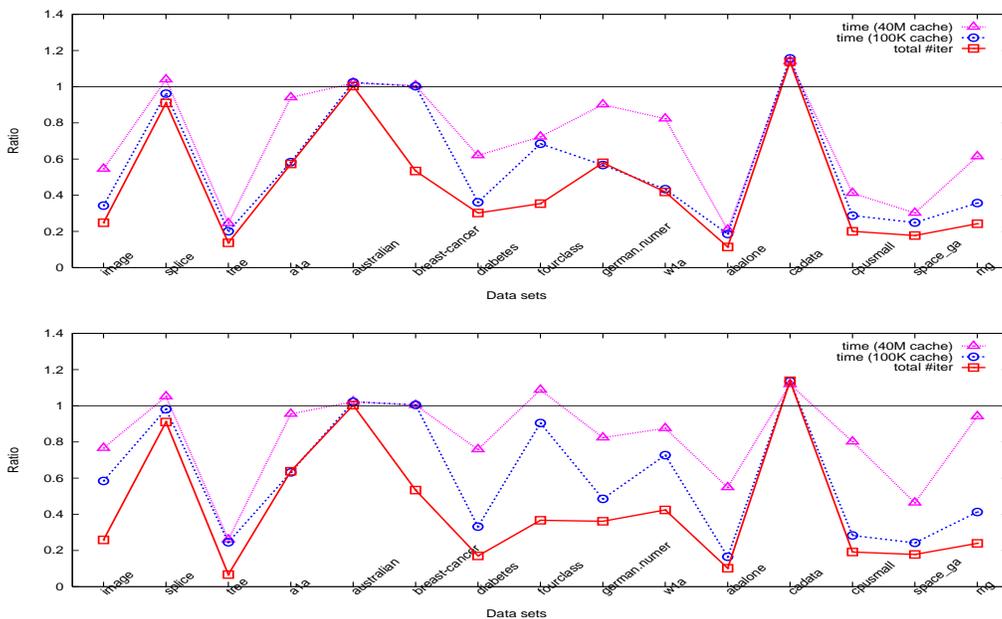


Figure 2: Iteration and time ratios between WSS 3 and 1 using the RBF kernel for the “final training” step (top: without shrinking, bottom: with shrinking).

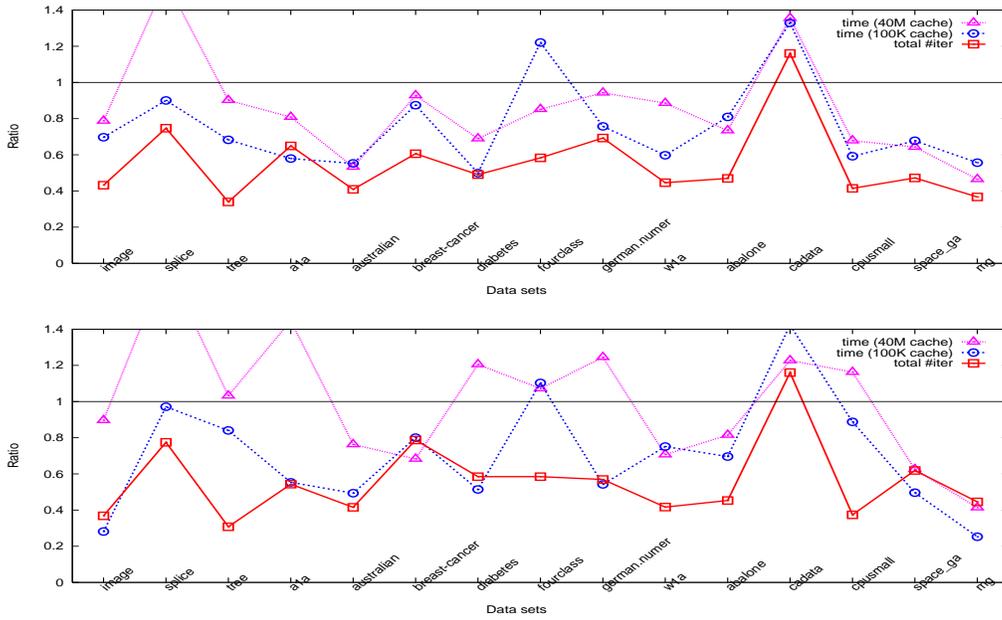


Figure 3: Iteration and time ratios between WSS 3 and 1 using the linear kernel for the “parameter selection” step (top: without shrinking, bottom: with shrinking).

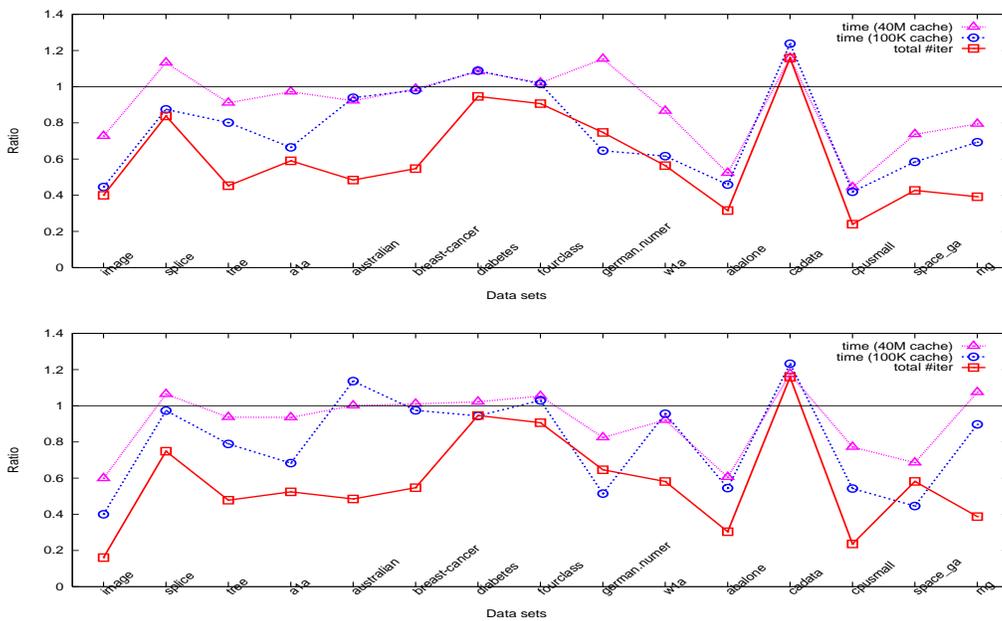


Figure 4: Iteration and time ratios between WSS 3 and 1 using the linear kernel for the “final training” step (top: without shrinking, bottom: with shrinking).

WORKING SET SELECTION FOR TRAINING SVMs

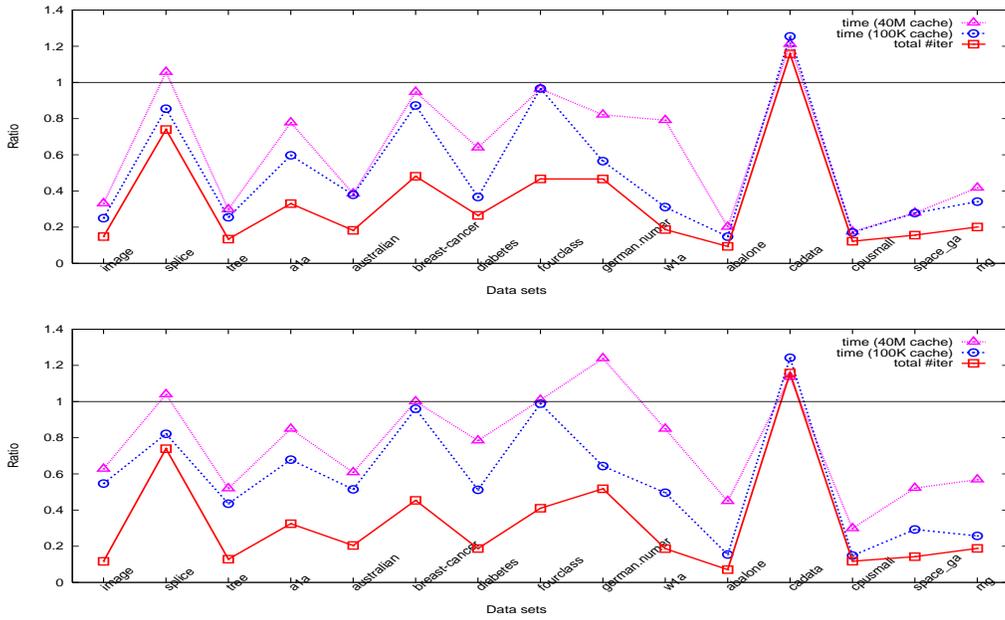


Figure 5: Iteration and time ratios between WSS 3 and 1 using the polynomial kernel for the “parameter selection” step (top: without shrinking, bottom: with shrinking).

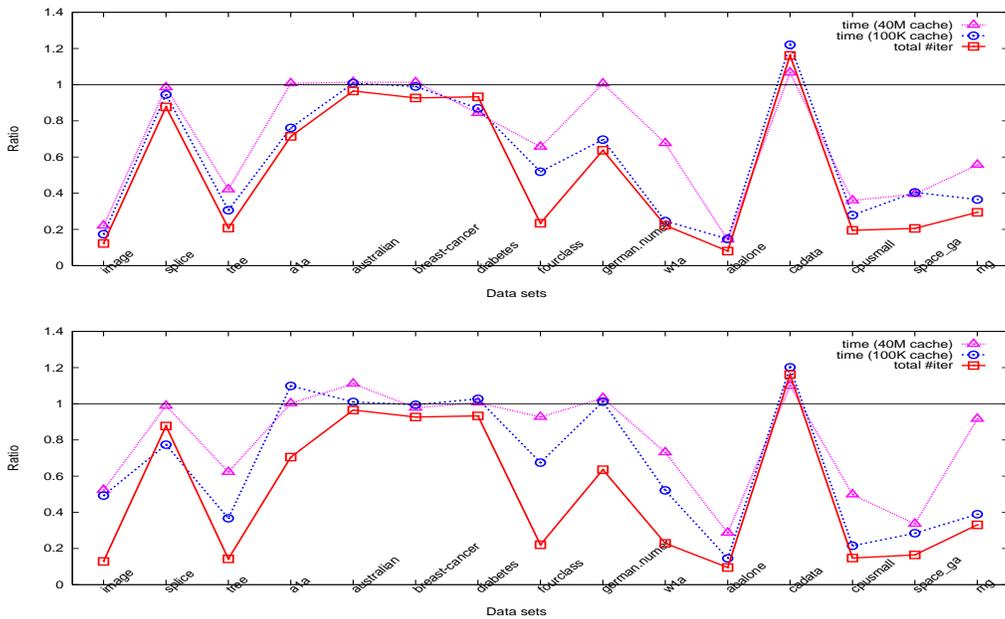


Figure 6: Iteration and time ratios between WSS 3 and 1 using the polynomial kernel for the “final training” step (top: without shrinking, bottom: with shrinking).

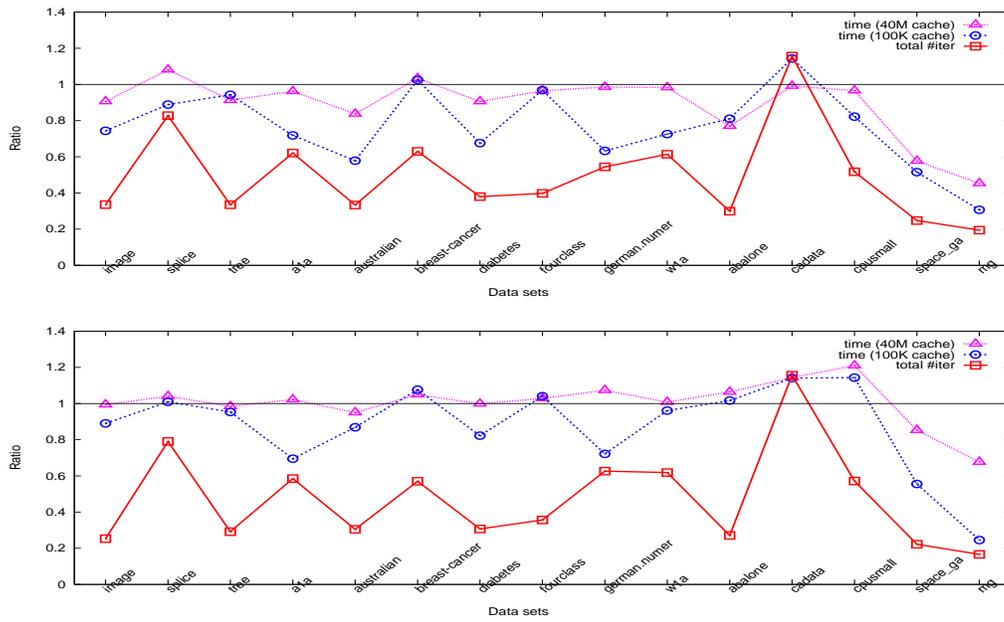


Figure 7: Iteration and time ratios between WSS 3 and 1 using the sigmoid kernel for the “parameter selection” step (top: without shrinking, bottom: with shrinking).

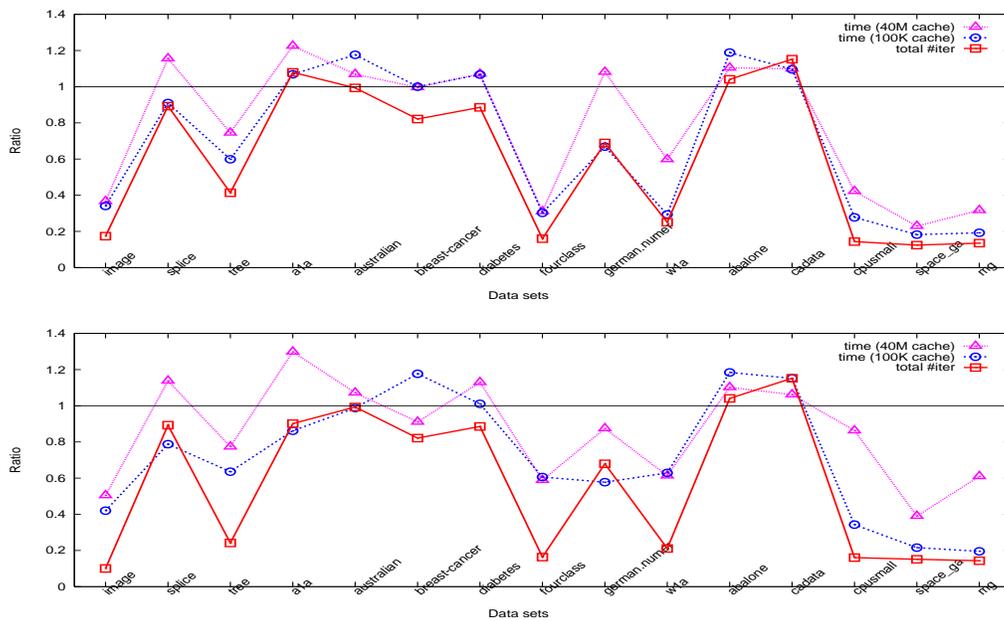


Figure 8: Iteration and time ratios between WSS 3 and 1 using the sigmoid kernel for the “final training” step (top: without shrinking, bottom: with shrinking).

Problem	#data	#feat.	RBF kernel				Linear kernel			
			Shrinking		No-Shrinking		Shrinking		No-Shrinking	
			Iter.	Time	Iter.	Time	Iter.	Time	Iter.	Time
a9a	32,561	123	0.73	0.92	0.75	0.93	0.86	0.92	0.88	0.95
w8a	49,749	300	0.48	0.72	0.50	0.81	0.47	0.90	0.53	0.79
IJCNN1	49,990	22	0.09	0.68	0.11	0.43	0.37	0.91	0.41	0.74
covtype*	100,000	54	0.37	0.90	0.37	0.76	0.19	0.59	0.22	0.52

Table 3: Large problems: Iteration and time ratios between WSS 3 and WSS 1 for the 16-point parameter selection. *: subset of a two-class data transformed from the original multi-class problem.

Since WSS 3 costs more than WSS 1, with (31),

$$\begin{aligned}
1 &\leq \frac{\text{time per Alg. 2 iteration (WSS 3, 100K cache)}}{\text{time per Alg. 2 iteration (WSS 1, 100K cache)}} \\
&\leq \frac{\text{time per Alg. 2 iteration (WSS 3, 40M cache)}}{\text{time per Alg. 2 iteration (WSS 1, 40M cache)}}.
\end{aligned}$$

This and (32) then imply (30). When shrinking is incorporated, the cost per iteration varies and (32) may not hold. Thus, though the relationship (30) in general still holds, there are more exceptions.

With the above analysis, our main observations and conclusions from Figures 1-8 are in the following:

1. Using WSS 3 significantly reduces the number of iterations. The reduction is more dramatic for the “parameter selection” step, where some points have slow convergence.
2. The new method is in general faster. Using a smaller cache gives better improvement. When the cache is not enough to store the whole kernel matrix, kernel evaluations are the main cost per iteration. Thus the time reduction is closer to the iteration reduction. This property hints that WSS 3 is useful on large-scale sets for which kernel matrices are too huge to be stored.
3. The implementation without shrinking gives better timing improvement than that with, even though they have similar iteration reduction. Shrinking successfully reduces the problem size and hence the memory use. Then similar to having enough cache, the time reduction does not match that of iterations due to the higher cost on selecting the working set per iteration. Therefore, results in Figures 1-8 indicate that with effective shrinking and caching implementations, it is difficult to have a new selection rule systematically surpassing WSS 1. The superior performance of WSS 3 thus makes important progress in training SVMs.

Next we experiment with large classification sets by a similar procedure. As the parameter selection is time consuming, we first use 10% training instances to identify a small region of good parameters. Then a 16-point search using the whole set is performed. The cache

size is 350M except 800M for covtype. We experiment with RBF and linear kernels. Table 3 gives iteration and time ratios of conducting the 16-point parameter selection. Similar to results for small problems, the number of iterations using WSS 3 is much smaller than that of using WSS 1. The training time of using WSS 3 is also shorter.

6. Maintaining Feasibility in Sub-problems for Working Set Selections

In Section 2, both the linear sub-problem (8) and quadratic sub-problem (11) do not require $\alpha^k + \mathbf{d}$ to be feasible. One may wonder if enforcing the feasibility gives a better working set and hence leads to faster convergence. In this situation, the quadratic sub-problem becomes

$$\begin{aligned} \text{Sub}(B) \equiv \min_{\mathbf{d}_B} & \quad \frac{1}{2} \mathbf{d}_B^T \nabla^2 f(\alpha^k)_{BB} \mathbf{d}_B + \nabla f(\alpha^k)_B^T \mathbf{d}_B \\ \text{subject to} & \quad \mathbf{y}_B^T \mathbf{d}_B = 0, \\ & \quad -\alpha_t^k \leq d_t \leq C - \alpha_t^k, \forall t \in B. \end{aligned} \tag{33}$$

For example, from some candidate pairs, Lai et al. (2003a,b) select the one with the smallest value of (33) as the working set. To check the effect of using (33), here we replace (11) in WSS 2 with (33) and compare it with the original WSS 2.

From (9), a nice property of using (33) is that $\text{Sub}(B)$ equals the decrease of the objective function f by moving from α^k to another feasible point $\alpha^k + \mathbf{d}$. In fact, once B is determined, (33) is also the sub-problem (2) used in Algorithm 1 to obtain α^{k+1} . Therefore, we use the same sub-problem for both selecting the working set and obtaining the next iteration α^{k+1} . One may think that such a selection method is better as it leads to the largest function value reduction while maintaining the feasibility. However, solving (33) is more expensive than (11) since checking the feasibility requires additional effort. To be more precise, if $B = \{i, j\}$, using $\hat{d}_j = -\hat{d}_i = y_j d_j = -y_i d_i$ and a derivation similar to (13), we now must minimize $\frac{1}{2} \bar{a}_{ij} \hat{d}_j^2 + b_{ij} \hat{d}_j$ under the constraints

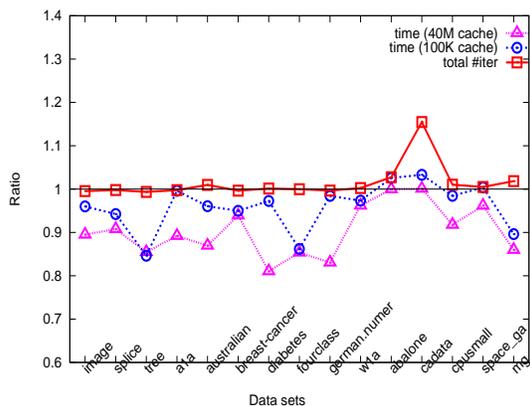
$$-\alpha_j^k \leq d_j = y_j \hat{d}_j \leq C - \alpha_j^k \text{ and } -\alpha_i^k \leq d_i = -y_i \hat{d}_j \leq C - \alpha_i^k. \tag{34}$$

As the minimum of the objective function happens at $-b_{ij}/\bar{a}_{ij}$, to have a solution satisfying (34), we multiply it by $-y_i$ and check the second constraint. Next, by $d_j = -y_i y_j d_i$, we check the first constraint. Equation (20) in WSS 3 is thus modified to

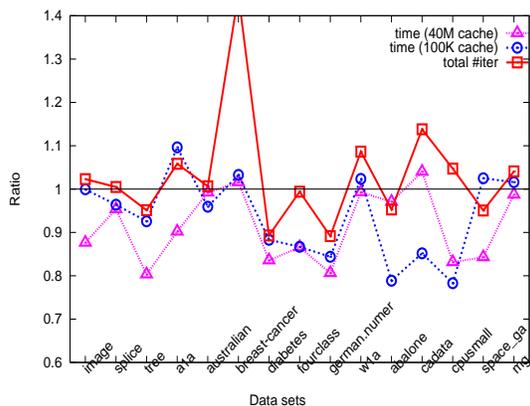
$$\begin{aligned} j \in \arg \min_t & \left\{ \frac{1}{2} \bar{a}_{it} \hat{d}_t^2 + b_{it} \hat{d}_t \mid t \in I_{\text{low}}(\alpha^k), -y_t \nabla f(\alpha^k)_t < -y_i \nabla f(\alpha^k)_i, \right. \\ & \left. \hat{d}_t = y_t \max(-\alpha_t^k, \min(C - \alpha_t^k, -y_t y_i \max(-\alpha_i^k, \min(C - \alpha_i^k, y_i b_{it}/\bar{a}_{it})))) \right\}. \end{aligned} \tag{35}$$

Clearly (35) requires more operations than (20).

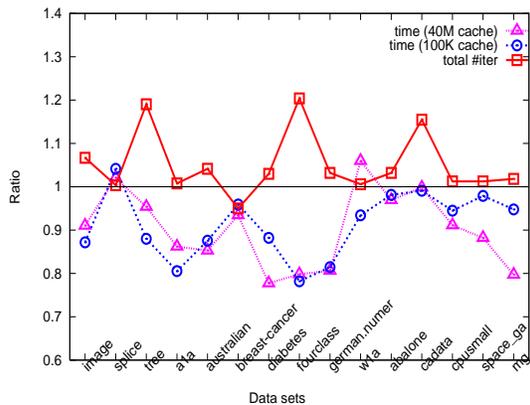
In this section, we prove that under some minor assumptions, in final iterations, solving (11) in WSS 2 is the same as solving (33). This result and experiments then indicate that there is no need to use the more sophisticated sub-problem (33) for selecting working sets.



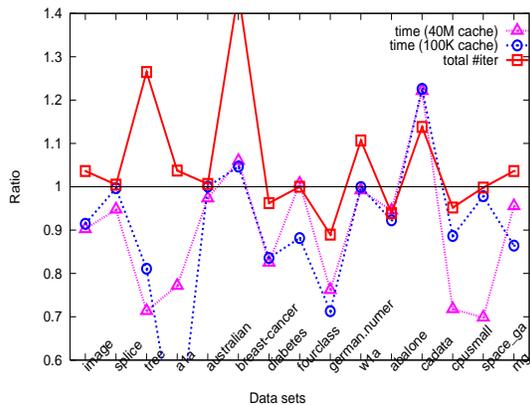
(a) The “parameter selection” step without shrinking



(b) The “final training” step without shrinking



(c) The “parameter selection” step with shrinking



(d) The “final training” step with shrinking

Figure 9: Iteration and time ratios between using (11) and (33) in WSS 2. Note that the ratio (y -axis) starts from 0.6 but not 0.

6.1 Solutions of (11) and (33) in final iterations

Theorem 7 Let $\{\alpha^k\}$ be the infinite sequence generated by the SMO-type decomposition method using WSS 2. Under the same assumptions of Theorem 6, there is \bar{k} such that for $k \geq \bar{k}$, WSS 2 returns the same working set by replacing (11) with (33).

Proof Since K is assumed to be positive definite, problem (1) has a unique optimal solution $\bar{\alpha}$. Using Theorem 4,

$$\lim_{k \rightarrow \infty} \alpha^k = \bar{\alpha}. \quad (36)$$

Since $\{\boldsymbol{\alpha}^k\}$ is an infinite sequence, Theorem 5 shows that $m(\bar{\boldsymbol{\alpha}}) = M(\bar{\boldsymbol{\alpha}})$. Hence we can define the following set

$$I' \equiv \{t \mid -y_t \nabla f(\bar{\boldsymbol{\alpha}})_t = m(\bar{\boldsymbol{\alpha}}) = M(\bar{\boldsymbol{\alpha}})\}.$$

As $\bar{\boldsymbol{\alpha}}$ is a non-degenerate point, from (27),

$$\delta \equiv \min_{t \in I'} (\bar{\alpha}_t, C - \bar{\alpha}_t) > 0.$$

Using 1) Eq. (36), 2) $\nabla f(\bar{\boldsymbol{\alpha}})_i = \nabla f(\bar{\boldsymbol{\alpha}})_j, \forall i, j \in I'$, and 3) Eq. (25) of Theorem 5, there is \bar{k} such that for all $k \geq \bar{k}$,

$$|\alpha_i^k - \bar{\alpha}_i| < \frac{\delta}{2}, \forall i \in I', \quad (37)$$

$$\frac{|-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j|}{K_{ii} + K_{jj} - 2K_{ij}} < \frac{\delta}{2}, \forall i, j \in I', K_{ii} + K_{jj} - 2K_{ij} > 0, \quad (38)$$

and

$$\text{all violating pairs come from } I'. \quad (39)$$

For any given index pair B , let $\text{Sub}_{(11)}(B)$ and $\text{Sub}_{(33)}(B)$ denote the optimal objective values of (11) and (33), respectively. If $\bar{B} = \{i, j\}$ is a violating pair selected by WSS 2 at the k th iteration, (37)-(39) imply that d_i and d_j defined in (15) satisfy

$$0 < \alpha_i^{k+1} = \alpha_i^k + d_i < C \text{ and } 0 < \alpha_j^{k+1} = \alpha_j^k + d_j < C. \quad (40)$$

Therefore, the optimal $\mathbf{d}_{\bar{B}}$ of (11) is feasible for (33). That is,

$$\text{Sub}_{(33)}(\bar{B}) \leq \text{Sub}_{(11)}(\bar{B}). \quad (41)$$

Since (33)'s constraints are stricter than those of (11), we have

$$\text{Sub}_{(11)}(B) \leq \text{Sub}_{(33)}(B), \forall B. \quad (42)$$

From WSS 3,

$$j \in \arg \min_t \{\text{Sub}_{(11)}(\{i, t\}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i\}.$$

With (41) and (42), this j satisfies

$$j \in \arg \min_t \{\text{Sub}_{(33)}(\{i, t\}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i\}.$$

Therefore, replacing (11) in WSS 3 with (33) does not affect the selected working set. \blacksquare

This theorem indicates that the two methods of working set selection in general lead to a similar number of iterations. As (11) does not check the feasibility, the implementation of using it should be faster.

6.2 Experiments

Under the framework WSS 2, we conduct experiments to check if using (11) is really faster than using (33). The same data sets in Section 5 are used under the same setting. For simplicity, we consider only the RBF kernel.

Similar to figures in Section 5, here Figure 9 presents iteration and time ratios between using (11) and (33):

$$\frac{\# \text{ iter. by using (11)}}{\# \text{ iter. by using (33)}}, \frac{\text{time by using (11) (100K cache)}}{\text{time by using (33) (100K cache)}}, \frac{\text{time by using (11) (40M cache)}}{\text{time by using (33) (40M cache)}}.$$

Without shrinking, clearly both approaches have very similar numbers of iterations. This observation is expected due to Theorem 7. Then as (33) costs more than (11) does, the time ratio is in general smaller than one. Especially when the cache is large enough to store all kernel elements, selecting working sets is the main cost and hence the ratio is lower.

With shrinking, in Figures 9(c) and 9(d), the iteration ratio is larger than one for several problems. Surprisingly, the time ratio, especially that of using a small cache, is even smaller than that without shrinking. In other words, (11) better incorporates the shrinking technique than (33) does. To analyze this observation, we check the number of removed variables along iterations, and find that (11) leads to more aggressive shrinking. Then the reduced problem can be stored in the small cache (100K), so kernel evaluations are largely saved. Occasionally the shrinking is too aggressive so some variables are wrongly removed. Then recovering from mistakes causes longer iterations.

Note that our shrinking implementation is by removing bounded elements not in the set (25). Thus, the smaller the interval $[M(\alpha^k), m(\alpha^k)]$ is, the more variables are shrunk. In Figure 10, we show the relationship between the maximal violation $m(\alpha^k) - M(\alpha^k)$ and iterations. Clearly using (11) reduces the maximal violation more quickly than using (33). A possible explanation is that (11) has less restriction than (33): In early iterations, if a set $B = \{i, j\}$ is associated with a large violation $-y_i \nabla f(\alpha^k)_i + y_j \nabla f(\alpha^k)_j$, then \mathbf{d}_B defined in (15) has large components. Hence though it minimizes the quadratic functions (9), $\alpha_B^k + \mathbf{d}_B$ is easily infeasible. To solve (33), one thus changes $\alpha_B^k + \mathbf{d}_B$ back to the feasible region as (35) does. As a reduced step is taken, the corresponding $\text{Sub}(B)$ may not be smaller than those of using other sets. On the other hand, (11) does not require $\alpha_B^k + \mathbf{d}_B$ to be feasible, so a large step is taken. The resulting $\text{Sub}(B)$ thus may be small enough so that B is selected. Therefore, using (11) tend to select working sets with large violations and hence may more quickly reduce the maximal violation.

Discussion here shows that (11) is better than (33). They lead to similar numbers of iterations, but the cost per iteration is less by using (11). Moreover, (11) better incorporates the shrinking technique.

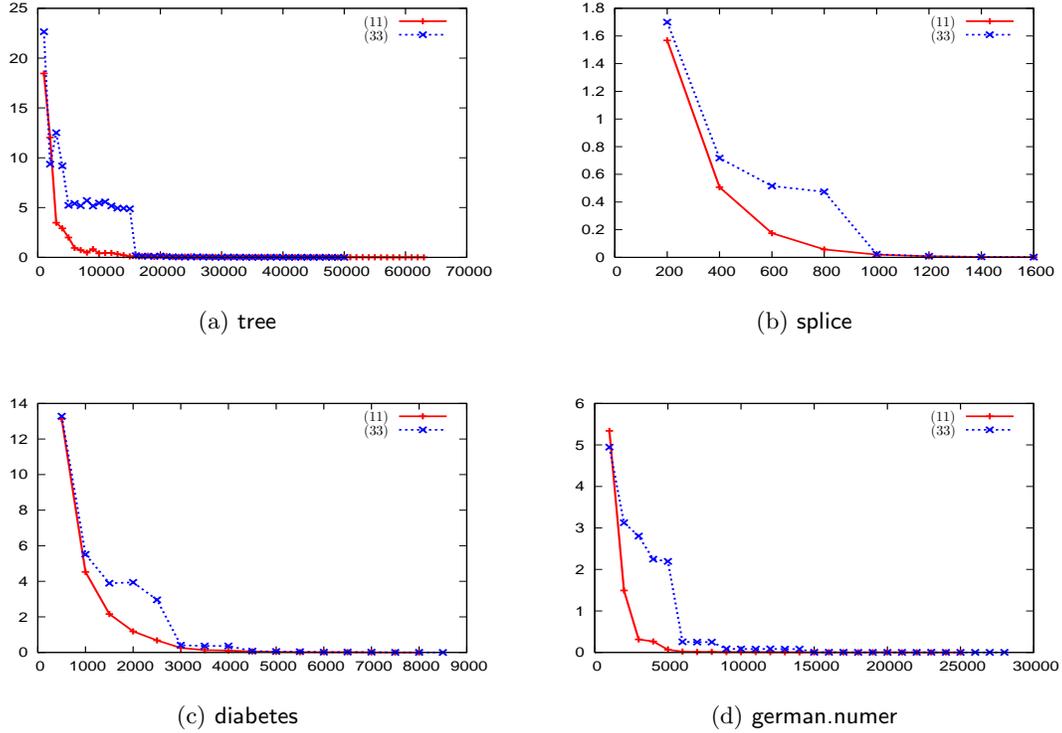


Figure 10: Iterations (x -axis) and maximal violations (y -axis) of using (11) and (33).

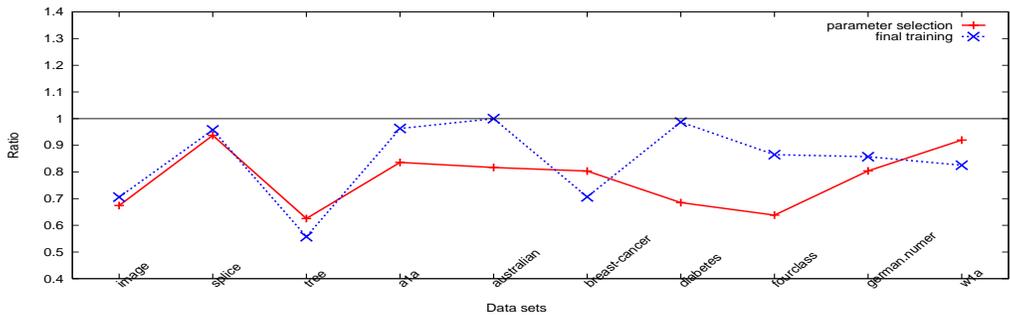
6.3 Sub-problems Using First Order Information

Under first order approximation, we can also modify the sub-problem (8) to the following form, which maintains the feasibility:

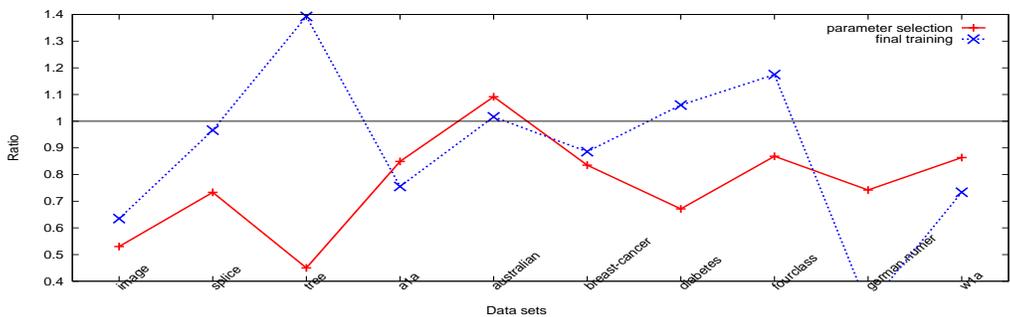
$$\begin{aligned}
 \text{Sub}(B) \equiv \min_{\mathbf{d}_B} \quad & \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B \\
 \text{subject to} \quad & \mathbf{y}_B^T \mathbf{d}_B = 0, \\
 & 0 \leq \alpha_i + d_i \leq C, i \in B.
 \end{aligned} \tag{43}$$

Section 2 discusses that a maximal violating pair is an optimal solution of $\min_{B:|B|=2} \text{Sub}(B)$, where $\text{Sub}(B)$ is (8). If (43) is used instead, Simon (2004) has shown an $O(l)$ procedure to obtain a solution. Thus the time complexity is the same as that of using (8).

Note that Theorem 7 does not hold for these two selection methods. In the proof, we use the small changes of α_i^k in final iterations to show that certain α_i^k never reaches bounds 0 and C . Then the sub-problem (2) to find $\boldsymbol{\alpha}^{k+1}$ is indeed the best sub-problem obtained in the procedure of working set selection. Now no matter (8) or (43) is used for selecting the working set, we still use (2) to find $\boldsymbol{\alpha}^{k+1}$. Therefore, we cannot link the small change between $\boldsymbol{\alpha}^k$ and $\boldsymbol{\alpha}^{k+1}$ to the optimal \mathbf{d}_B in the procedure of working set selection. Without an interpretation like Theorem 7, the performance difference between using (8) and (43) remains unclear and is a future research issue.



(a) RBF kernel



(b) Linear kernel

Figure 11: Iteration ratios between using two selection methods: checking all $\binom{l}{2}$ pairs and WSS 2. Note that the ratio (y -axis) starts from 0.4 but not 0.

7. Discussion and Conclusions

In Section 2, the selection (10) of using second order information may involve checking $\binom{l}{2}$ pairs of indices. This is not practically viable, so in WSS 2 we heuristically fix $i \in \arg m(\alpha^k)$ and examine $O(l)$ sets to find j . It is interesting to see how well this heuristic performs and whether we can make further improvements. By running the same small classification problems used in Section 6, Figure 11 presents the iteration ratio between using two selection methods:

$$\frac{\# \text{ iter. by Alg. 2 and checking } \binom{l}{2} \text{ pairs}}{\# \text{ iter. by Alg. 2 and WSS 2}}$$

We do not use shrinking and consider both RBF and linear kernels. Figure 11 clearly shows that a full check of all index pairs causes fewer iterations. However, as the average of ratios for various problems is between 0.7 and 0.8, this selection reduces iterations of using WSS 2 by only 20% to 30%. Therefore, WSS 2, an $O(l)$ procedure, successfully returns a working set nearly as good as that by an $O(l^2)$ procedure. In other words, the $O(l)$ sets heuristically considered in WSS 2 are among the best in all $\binom{l}{2}$ candidates.

Experiments in this paper fully demonstrate that using the proposed WSS 2 (and hence WSS 3) leads to faster convergence (i.e., fewer iterations) than using WSS 1. This result is reasonable as the selection based on second order information better approximates the objective function in each iteration. However, this argument explains only the behavior per iteration, but not the global performance of the decomposition method. A theoretical study showing that the proposed selection leads to better convergence rates is a difficult but interesting future issue.

In summary, we have proposed a new and effective working set selection WSS 3. The SMO-type decomposition method using it asymptotically converges and satisfies other useful theoretical properties. Experiments show that it is better than a commonly used selection WSS 1, in both the training time and iterations.

WSS 3 has replaced WSS 1 in the software LIBSVM (after version 2.8).

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grant NSC 93-2213-E-002-030.

Appendix A. WSS 1 Solves Problem (7): the Proof

For any given $\{i, j\}$, we can substitute $\hat{d}_i \equiv y_i d_i$ and $\hat{d}_j \equiv y_j d_j$ to (8), so the objective function becomes

$$(-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j) \hat{d}_j. \tag{44}$$

As $d_i = d_j = 0$ is feasible for (8), the minimum of (44) is zero or a negative number. If $-y_i \nabla f(\boldsymbol{\alpha}^k)_i > -y_j \nabla f(\boldsymbol{\alpha}^k)_j$, using the condition $\hat{d}_i + \hat{d}_j = 0$, the only possibility for (44) to be negative is $\hat{d}_j < 0$ and $\hat{d}_i > 0$. From (3), (8b), and (8c), this corresponds to $i \in I_{\text{up}}(\boldsymbol{\alpha}^k)$ and $j \in I_{\text{low}}(\boldsymbol{\alpha}^k)$. Moreover, the minimum occurs at $\hat{d}_j = -1$ and $\hat{d}_i = 1$. The situation of $-y_i \nabla f(\boldsymbol{\alpha}^k)_i < -y_j \nabla f(\boldsymbol{\alpha}^k)_j$ is similar.

Therefore, solving (7) is essentially the same as

$$\begin{aligned} & \min \left\{ \min(y_i \nabla f(\boldsymbol{\alpha}^k)_i - y_j \nabla f(\boldsymbol{\alpha}^k)_j, 0) \mid i \in I_{\text{up}}(\boldsymbol{\alpha}^k), j \in I_{\text{low}}(\boldsymbol{\alpha}^k) \right\} \\ & = \min(-m(\boldsymbol{\alpha}^k) + M(\boldsymbol{\alpha}^k), 0). \end{aligned}$$

Hence, if there are violating pairs, the maximal one solves (7).

Appendix B. Pseudo Code of Algorithm 2 and WSS 3

B.1 Main Program (Algorithm 2)

Inputs:

```

y:   array of {+1, -1}: class of the i-th instance
Q:   Q[i][j] = y[i]*y[j]*K[i][j]; K: kernel matrix
len: number of instances

```

```
// parameters
```

```

eps = 1e-3 // stopping tolerance
tau = 1e-12

// main routine
initialize alpha array A to all zero
initialize gradient array G to all -1

while (1) {
  (i,j) = selectB()
  if (j == -1)
    break

  // working set is (i,j)
  a = Q[i][i]+Q[j][j]-2*y[i]*y[j]*Q[i][j]
  if (a <= 0)
    a = tau
  b = -y[i]*G[i]+y[j]*G[j]

  // update alpha
  oldAi = A[i], oldAj = A[j]
  A[i] += y[i]*b/a
  A[j] -= y[j]*b/a

  // project alpha back to the feasible region
  sum = y[i]*oldAi+y[j]*oldAj
  if A[i] > C
    A[i] = C
  if A[i] < 0
    A[i] = 0
  A[j] = y[j]*(sum-y[i]*A[i])
  if A[j] > C
    A[j] = C
  if A[j] < 0
    A[j] = 0
  A[i] = y[i]*(sum-y[j]*A[j])

  // update gradient
  deltaAi = A[i] - oldAi, deltaAj = A[j] - oldAj
  for t = 1 to len
    G[t] += Q[t][i]*deltaAi+Q[t][j]*deltaAj
}

```

B.2 Working Set Selection Subroutine (WSS 3)

```

// return (i,j)
procedure selectB
  // select i
  i = -1
  G_max = -infinity
  G_min = infinity
  for t = 1 to len {

```

```

    if (y[t] == +1 and A[t] < C) or
        (y[t] == -1 and A[t] > 0) {
        if (-y[t]*G[t] >= G_max) {
            i = t
            G_max = -y[t]*G[t]
        }
    }
}

// select j
j = -1
obj_min = infinity
for t = 1 to len {
    if (y[t] == +1 and A[t] > 0) or
        (y[t] == -1 and A[t] < C) {
        b = G_max + y[t]*G[t]
        if (-y[t]*G[t] <= G_min)
            G_min = -y[t]*G[t]
        if (b > 0) {
            a = Q[i][i]+Q[t][t]-2*y[i]*y[t]*Q[i][t]
            if (a <= 0)
                a = tau
            if (-(b*b)/a <= obj_min) {
                j = t
                obj_min = -(b*b)/a
            }
        }
    }
}

if (G_max-G_min < eps)
    return (-1,-1)

return (i,j)
end procedure

```

References

- R. R. Bailey, E. J. Pettit, R. T. Borochoff, M. T. Manry, and X. Jiang. Automatic recognition of usgs land use/cover categories using statistical and neural networks classifiers. In *SPIE OE/Aerospace and Remote Sensing*, Bellingham, WA, 1993. SPIE.
- C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA, 1998. Available at <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

- Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Pai-Hsuen Chen, Rong-En Fan, and Chih-Jen Lin. A study on SMO-type decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 2006. URL <http://www.csie.ntu.edu.tw/~cjlin/papers/generalSMO.pdf>. To appear.
- C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.
- Tin Kam Ho and Eugene M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 880–885, Vienna, Austria, August 1996.
- Don Hush and Clint Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51:51–71, 2003. URL http://www.c3.lanl.gov/~dhush/machine_learning/svm_decomp.ps.
- Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.
- S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt’s SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.
- D. Lai, N. Mani, and M. Palaniswami. Increasing the step of the Newtonian decomposition method for support vector machines. Technical Report MECSE-29-2003, Dept. Electrical and Computer Systems Engineering Monash University, Australia, 2003a.
- D. Lai, N. Mani, and M. Palaniswami. A new method to select working sets for faster training for support vector machines. Technical Report MESCE-30-2003, Dept. Electrical and Computer Systems Engineering Monash University, Australia, 2003b.
- Chih-Jen Lin. Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science, National Taiwan University, 2001a. URL <http://www.csie.ntu.edu.tw/~cjlin/papers/linearconv.pdf>.
- Chih-Jen Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, 2001b. URL <http://www.csie.ntu.edu.tw/~cjlin/papers/conv.ps.gz>.
- Chih-Jen Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13(1):248–250, 2002. URL <http://www.csie.ntu.edu.tw/~cjlin/papers/q2conv.pdf>.
- D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J., 1994. Data available at <http://www.ncc.up.pt/liacc/ML/statlog/datasets.html>.
- E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of CVPR’97*, pages 130–136, New York, NY, 1997. IEEE.

- Laura Palagi and Marco Sciandrone. On the convergence of a modified version of SVM^{light} algorithm. *Optimization Methods and Software*, 20(2-3):315–332, 2005.
- J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.
- Danil Prokhorov. IJCNN 2001 neural network competition. Slide presentation in IJCNN'01, Ford Research Laboratory, 2001. http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf.
- B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.
- Hans Ulrich Simon. On the complexity of working set selection. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT 2004)*, 2004.

New Horn Revision Algorithms

Judy Goldsmith

*Department of Computer Science
University of Kentucky
773 Anderson Tower
Lexington, KY 40506-0046*

GOLDSMIT@CS.UKY.EDU

Robert H. Sloan

*University of Illinois at Chicago
Department of Computer Science
851 South Morgan Street, Room 1120
Chicago, IL 60607-7053*

SLOAN@UIC.EDU

Editor: Stefan Wrobel

Abstract

A revision algorithm is a learning algorithm that identifies the target concept, starting from an initial concept. Such an algorithm is considered efficient if its complexity (in terms of the measured resource) is polynomial in the syntactic distance between the initial and the target concept, but only polylogarithmic in the number of variables in the universe. We give efficient revision algorithms in the model of learning with equivalence and membership queries. The algorithms work in a general revision model where both deletion and addition revision operators are allowed. In this model one of the main open problems is the efficient revision of Horn formulas. Two revision algorithms are presented for special cases of this problem: for depth-1 acyclic Horn formulas, and for definite Horn formulas with unique heads.

Keywords: theory revision, Horn formulas, query learning, exact learning, computational learning theory

1. Introduction

Computationally efficient learnability has been studied in the past two decades from many angles. For example, both the PAC and query learning models have been studied, and complexity has been variously measured in terms of sample size, the number of queries, and running time. Attribute-efficient learning algorithms are required to be efficient (polynomial) in the number of relevant variables, and “super-efficient” (polylogarithmic) in the total number of variables (Blum et al., 1995; Bshouty and Hellerstein, 1998).

A related notion, *efficient revision algorithms*, has been studied in machine learning, where various approaches to building systems have been considered (see, e.g., Koppel et al., 1994; Lamma et al., 2003; Ourston and Mooney, 1994; Richards and Mooney, 1995; Towell and Shavlik, 1993). Efficient revision algorithms have received some attention in learning theory as well. A revision algorithm is applied in a situation where learning does not start from scratch, but there is an initial concept available, which is a reasonable approximation of the target concept. The standard example is an initial version of an expert system provided by a domain expert. The efficiency criterion in this

case is to be efficient (polynomial) in the *distance* from the initial concept to the target (whatever distance means; we get back to this in a minute), and to be “super-efficient” (polylogarithmic) in the total size of the initial formula. Again, it is argued that this is a realistic requirement, since, for many complex concepts, the only hope of learning those concepts is if a reasonably good initial approximation is available.

The notion of distance usually considered for efficient revision is a syntactic one: the number of edit operations that need to be applied to the initial representation in order to get a representation of the target. The particular edit operations considered depend on the concept class. Intuitively, attribute-efficient learning is a special case of efficient revision, when the initial concept has an empty representation. In machine learning, the study of revision algorithms is referred to as theory revision; more detailed references to the literature are given in Wrobel’s overviews of theory revision (Wrobel, 1994, 1995) and also in our recent papers (Goldsmith et al., 2002, 2004b).

The theoretical study of revision algorithms was initiated by Mooney (1995) in the PAC framework, and additional theoretical work was done by Greiner (1999a,b). We have studied revision algorithms in the model of learning with equivalence and membership queries (Goldsmith et al., 2002, 2004b) and in the mistake-bound model (Sloan et al., 2003).

It is a general observation both in practice and in theory that those edit operations which delete something from the initial representation are easier to handle than those which add something to it. We have obtained efficient revision algorithms for monotone¹ Disjunctive Normal Form (DNF) with a bounded number of terms when both deletion and addition type revisions are allowed, but for the practically important case of Horn formulas we found an efficient revision algorithm only for the deletions-only model. We also showed that efficient revision of general (or even monotone) DNF is not possible, even in the deletions-only model. Finding an efficient revision algorithm for Horn formulas in the general revision model (deletions and additions) emerged as perhaps the main open problem posed by our previous work on revision algorithms. One of the two results presented here extends that of Doshi (2003), who gave a revision algorithm for a special case of Horn sentences he called “unique explanations,” which in the terminology presented below would be the special case of depth-1 acyclic Horn sentences where the heads must all be distinct, every clause must have a head, and the heads cannot be revised. The result we give in Section 3 removes all of his restrictions concerning the heads.

1.1 Revision with Queries

In this paper, we consider revision in *query-based* learning models, in particular, in the standard model of learning with *membership* and *equivalence* queries, denoted by MQ and EQ (Angluin, 1988). This is a very well-studied model (e.g., Angluin, 1987b, 1988; Angluin et al., 1992; Auer and Long, 1999; Bshouty and Hellerstein, 1998; Blum et al., 2004; Bshouty, 1995), nearly as much so as PAC learning. In an equivalence query, the learning algorithm proposes a *hypothesis*, that is, a theory h , and the answer depends on whether $h = c$, where c is the target theory. If so, the answer is “correct”, and the learning algorithm has succeeded in its goal of exact identification of the target theory. Otherwise, the answer is a *counterexample*: any instance x such that $c(x) \neq h(x)$. In a membership query, the learning algorithm gives an instance x , and the answer is either 1 or 0, depending on $c(x)$.

1. A propositional logic formula is *monotone* if it contains no negations.

The *query complexity* of a learning algorithm is the number of queries it asks. Note that the query complexity is a lower bound on the running time. For running time, we do *not* count the time required to answer the queries. From a formal, theoretical point of view, we assume that there are two oracles, one each to answer membership and equivalence queries. In practice, membership queries would need to be answered by a domain expert, and equivalence queries could either be answered by a domain expert, or by using the hypothesis and waiting for evidence of an error in classification.

One scenario for practical applications is that one starts with an initial theory and a set of (counter)examples, for which the initial theory gives an incorrect classification. The goal then is to find a small modification of the initial theory that is consistent with the examples. In this setup, one can simulate an equivalence query by running through the examples. If we find a counterexample to the current hypothesis, then we continue the simulation of the algorithm. Otherwise, we terminate the learning process with the current hypothesis serving as our final revised theory. In this way, an efficient equivalence and membership query algorithm can be turned into an efficient practical revision algorithm.

Perhaps the most common case for practical applications of theory revision is to fix an initial theory that is provided by an expert. It is reasonable to hope that the expert is able to answer further queries about the classification of new instances. Consider the following case: Expert oncologist Dr. Jones is cooperating with the local computer scientists to build a model of foobaric cancer. She gives long answers to the knowledge engineers' initial open-ended questions, and countless shorter answers as they build and refine their model. These shorter questions are membership questions: "If the patient has this complex of symptoms, do you diagnose foobaric cancer?"

Finally, in the model validation phase of the work, the knowledge engineers and computer scientists proudly present scenarios and diagnoses. And Dr. Jones shakes her head and says, "No, that's not right at all. Your system will give the wrong diagnosis in these settings; reliance on this symptom is a red herring."

These latter responses are equivalence queries, complete with counterexamples.

As an aside, even theory revision via queries for formal languages may have some application. Consider Professor Doe, who is teaching, say, Automata and Formal Languages. Her difficult student presents her with an incorrect finite automaton, and demands proof that it is incorrect. She provides a counterexample, some string that the presented automaton misclassifies. It becomes clear that the student has misunderstood the problem. String by string, he queries her about membership in the desired regular language, offering periodic updates to his automaton until either it is correct, or Professor Doe discovers a prior appointment.

Note that an efficient revision algorithm is clearly in the student's best interest in this case.

1.2 Classes of Horn Formulas Considered

Horn revision is the problem that most practical theory revision systems address. It is to be noted here that the notions of learning and revising Horn formulas are open to interpretation, as discussed by Goldsmith et al. (2004b); the kind of learnability result that we wish to extend to revision in this paper is that of Angluin et al. (1992) for propositional Horn formulas.

In this paper we present results for the revision problem outlined above: the revision of Horn formulas in the general revision model allowing both deletions and additions (more precise defi-

nitions are given in Section 2). We use the model of learning with membership and equivalence queries.

We show that one can revise two subclasses of Horn formulas with respect to both additions and deletions of variables. The new algorithms make use of our previous, deletions-only revision algorithm for Horn formulas (Goldsmith et al., 2004b) and new techniques which could be useful for the general question as well.

1.2.1 DEPTH-1 ACYCLIC HORN

Logic programming theories are often presented as Horn theories. Each clause with a head, or nonnegated variable, is interpreted as a potential justification for making the head variable true in some model of the program. These clauses are also called “definite”.

In computing stable models of logic programs, it is simplest if the logic programs are stratified (Apt et al., 1988; Chandra and Harel, 1985; Van Gelder, 1988), or acyclic (Angluin, 1987a). One begins by setting all “facts,” or heads without bodies, to true.² Then iteratively, one sets all consequences of the current true variables to true.

At each iteration, one considers only definite clauses, and only those clauses whose heads do not appear in the currently-true variables and all of whose variables are already true. These collections of clauses, or strata of a program, are themselves depth-1 Horn theories. We begin by focusing on theory revision for these simple theories.

One of our main results, Theorem 5, shows that this class can be revised using $O(\text{dist}(\varphi, \psi) \cdot m^3 \cdot \log n)$ queries, where n is the number of variables, φ is the m -clause initial formula, ψ is the target formula, and dist is the revision distance, which will be defined formally in Section 2.

1.2.2 DISTINCT HEADS/UNIQUE EXPLANATIONS

In life, and in many Horn theories, there may be multiple explanations of something, or Horn clauses with the same head. Another simplification to Horn theories, other than considering individual strata, is to consider theories that provide unique explanations for each variable; that is, theories where clauses each have a distinct head. As in the stratified theories, this allows model-building to be accomplished in one pass through the theory. [Note that this definition of “unique explanation” is simpler than that of Doshi (2003). We also refer to such theories as having “distinct heads.”]

But even such simple theories are subject to revision. The expert who provides a theory may fudge on explanations, including unnecessary preconditions or omitting necessary ones. Thus, our second topic in this paper is revision with queries for theories consisting of unique explanations.

We also give a revision algorithm for definite Horn formulas with distinct heads, meaning that no variable ever occurs as the head of more than one Horn clause. For this class, we revise with query complexity $O(m^4 + \text{dist}(\varphi, \psi) \cdot (m^3 + \log n))$, where again φ is the initial formula and ψ is the target function (Theorem 8).

1.3 Overview of the Rest of the Paper

Preliminaries are given in Section 2, Horn formula revisions in Sections 3 and 4, and open questions in Section 5.

2. Acyclic Horn formulas have also been studied from various other points of view, including learning (Angluin, 1987a; Arimura, 1997) and computational aspects (Hammer and Kogan, 1995).

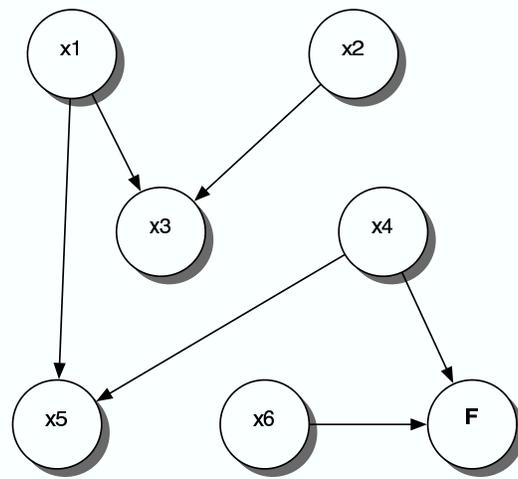


Figure 1: Graph of the Horn formula ϕ given by (1).

2. Preliminaries

We use standard notions from propositional logic such as variable, literal, term (or conjunction), clause (or disjunction), etc. The set of variables for n -variable formulas and functions is $X_n = \{x_1, \dots, x_n\}$. (In this paper, n will always be the total number of variables.) *Instances* or *vectors* are elements $\mathbf{x} \in \{0, 1\}^n$. In the vocabulary of propositional logic, an instance (or vector) is a model for the target theory. When convenient we treat \mathbf{x} as a subset of $[n]$ or X_n , corresponding to the components, resp. the variables, which are set to true in \mathbf{x} . Given a set $Y \subseteq [n] = \{1, \dots, n\}$, we write $\chi_Y = (\alpha_1, \dots, \alpha_n) \in \{0, 1\}^n$, where $\alpha_i = 1$ if $i \in Y$ and $\alpha_i = 0$ otherwise, for the characteristic vector of Y . We write $\mathbf{x} = (x_1, \dots, x_n) \leq \mathbf{y} = (y_1, \dots, y_n)$ if $x_i \leq y_i$ for every $i = 1, \dots, n$.

A *Horn clause* is a disjunction with at most one unnegated variable; we will usually think of it as an implication and call the clause's unnegated variable its *head*, and its negated variables its *body*. We write $\text{body}(c)$ and $\text{head}(c)$ for the body and head of c , respectively. When convenient, we treat $\text{body}(c)$ as the vector with 1's in the positions where $\text{body}(c)$ has variables. A Horn clause with an unnegated variable is called *definite* (or positive). If a definite clause contains only one variable, then that clause is called a *fact*. We will consider clauses with no unnegated variables to have head \mathbf{F} , and will sometimes write them as $(\text{body} \rightarrow \mathbf{F})$.

A *Horn formula* is a conjunction of Horn clauses. A Horn formula is definite if all its clauses are definite. A Horn formula has *unique heads* if no two clauses have the same head.

We define the *graph* of a Horn formula to be a directed graph on the variables together with \mathbf{F} , with an edge from variable u to variable v (resp. \mathbf{F}) iff there is a clause with head v (resp. \mathbf{F}) having u in its body. A Horn formula is *acyclic* if its graph is acyclic; the *depth* of an acyclic Horn formula is the maximum path length in its graph (Angluin, 1987a).

For example, the Horn formula

$$\phi = (x_1 \wedge x_2 \rightarrow x_3) \wedge x_2 \wedge (x_1 \wedge x_4 \rightarrow x_5) \wedge (x_4 \wedge x_6 \rightarrow \mathbf{F}) \quad (1)$$

is depth-1 acyclic. Its graph, shown in Figure 1, has the edges (x_1, x_3) , (x_2, x_3) , (x_1, x_5) , (x_4, x_5) , (x_4, \mathbf{F}) , and (x_6, \mathbf{F}) and this graph is acyclic with depth 1.

If \mathbf{x} satisfies the body of Horn clause c , considered as a term, we say \mathbf{x} *covers* c . Notice that \mathbf{x} *falsifies* c if and only if \mathbf{x} covers c and $\text{head}(c) \notin \mathbf{x}$. (By definition, $\mathbf{F} \notin \mathbf{x}$.)

For Horn clause body b (or any monotone term) and vector \mathbf{x} , we use $b \cap \mathbf{x}$ for the monotone term that has those variables of b that correspond to 1's in \mathbf{x} . As an example, $x_1x_4 \cap 1100 = x_1$.

We use the standard model of membership and equivalence queries (with counterexamples), denoted by MQ and EQ (Angluin, 1988). In an equivalence query, the learning algorithm proposes a *hypothesis*, a formula h , and the answer depends on whether $h \equiv c$, where c is the target formula. If so, the answer is “correct”, and the learning algorithm has succeeded in its goal of exact identification of the target concept. Otherwise, the answer is a *counterexample*, any instance \mathbf{x} such that $c(\mathbf{x}) \neq h(\mathbf{x})$. If \mathbf{x} is a counterexample and $c(\mathbf{x}) = 1$ and $h(\mathbf{x}) = 0$, then we refer to \mathbf{x} as a positive counterexample, and otherwise a negative counterexample.

2.1 Revision

The *revision distance* between a formula ϕ and a concept C is defined to be the minimum number of applications of a specified set of syntactic revision operators to ϕ needed to obtain a formula for C . The revision operators may depend on the concept class one is interested in. Usually, a revision operator can either be *deletion-type* or *addition-type*.

For disjunctive or conjunctive normal forms (including Horn sentences), the deletion operation can be formulated as *fixing an occurrence of a variable* in the formula to a constant. In the *general model*, studied in this paper, we also allow additions. The addition operation is to *add a new literal to one of the terms or clauses of the formula*. (Adding a new literal to open up a new clause or term would be an even more general addition-type operator, which we have not considered so far. Note that in Algorithm 2, while we “add clauses” to a hypothesis, these are always clauses that are in the given formula but not yet in the hypothesis.) In the algorithms given in this paper, the new literals must be added to the body of a clause.

We use $\text{dist}(\phi, \psi)$ to denote the revision distance from ϕ to ψ whenever the revision operators are clear from context. In general, the distance is not symmetric.

A *revision algorithm* for a formula ϕ has access to membership and equivalence oracles for an unknown target concept and must return some representation of the target concept. Our goal is to find revision algorithms whose query complexity is polynomial in $d = \text{dist}(\phi, \psi)$, but at most *polylogarithmic* in n , the number of variables in the universe. For DNF (resp. CNF) formulas, we allow polynomial dependence on the number of terms (resp. clauses) in ϕ ; it is impossible to do better even for arbitrary monotone DNF in the deletions-only model of revision (Goldsmith et al., 2002).

We state only query bounds in this paper; all our revision algorithms are computable in polynomial time, given the appropriate oracles.

2.2 Binary Search for New Variables

Our revision algorithms use a kind of binary search, of a general kind often used in learning algorithms involving membership queries, presented as Algorithm 1. The starting points of our binary search are two instances, a negative instance **neg** and a positive instance **pos** such that **pos** < **neg**. The algorithm returns a variable v that is critical in the sense that there is a (possibly empty) set S of variables from **neg** \ **pos** such that **neg** modified by setting the variables in S to 0 is still a negative instance, but additionally setting v to 0 creates a positive instance.

Algorithm 1 BINARYSEARCH(**neg**, **pos**).

Require: MQ(**neg**) == 0 and MQ(**pos**) == 1 and **pos** < **neg**

```

1: neg0 := neg
2: while neg and pos differ in more than 1 position do
3:   Partition neg \ pos into approximately equal-size sets  $d_1$  and  $d_2$ .
4:   Put mid := neg with positions in  $d_1$  switched to 1
5:   if MQ(mid) == 0 then
6:     neg := mid
7:   else
8:     pos := mid
9:   end if
10: end while
11:  $v :=$  the one variable on which pos and neg differ
12: return  $v$ 

```

3. Depth-1 Acyclic Horn Formulas

We show here how to revise depth-1 acyclic Horn formulas. Depth-1 acyclic Horn formulas are precisely those where each variable that occurs as a head either occurs as a fact (the head of an empty-bodied clause) or never occurs in the body of any clause. Notice that such formulas are a class of unate CNF: variables that occur as facts are the only variables that can appear both negated and unnegated, and we can always rewrite any Horn formula with facts to a logically equivalent Horn formula where those fact variables do not appear in any clause body by using resolution. For example, ϕ in Equation 1 is equivalent to

$$(x_1 \rightarrow x_3) \wedge x_2 \wedge (x_1 \wedge x_4 \rightarrow x_5) \wedge (x_4 \wedge x_6 \rightarrow \mathbf{F}).$$

Previously we gave a revision algorithm for unate DNF (which would dualize to unate CNF) that was presented as being able to revise specifically two clauses (Goldsmith et al., 2002). It would generalize to an algorithm whose query complexity is exponential in the number of clauses. Here we give an algorithm for an important subclass of unate CNF that is polynomial in the number of clauses.

In the following subsection we give the algorithm and its analysis; then in Section 3.2 we give an example run of the algorithm. The reader may find it helpful to switch back and forth between the two subsections.

3.1 Algorithm and Analysis

The general idea of the algorithm is to maintain a one-sided hypothesis, in the sense that all equivalence queries using the hypothesis must return negative counterexamples until the hypothesis is correct.

Each negative counterexample can be associated with one particular head of the target clause, or else with a headless target clause. We do this with a negative counterexample \mathbf{x} as follows.

Let us call those variables that occur as the head of a clause of the initial formula *head variables*. For a head variable v and instance \mathbf{x} , we will use the notation \mathbf{x}^v to refer to \mathbf{x} modified by setting all head variables *other than* v to 1. Note that \mathbf{x}^v cannot falsify any clause with a head other than v .

Since v will normally be the head of a Horn clause and we use \mathbf{F} to denote the “head” of a headless Horn clause, we will use $\mathbf{x}^{\mathbf{F}}$ to denote \mathbf{x} modified to set *all* head variables to 1.

We will implicitly use the following fact often in our analysis of our algorithm.

Proposition 1 *Let h be either a variable or \mathbf{F} . If \mathbf{x}^h falsifies a clause of the target depth-1 acyclic Horn formula with head h , then \mathbf{x} also falsifies that clause.*

Proof Consider target clause $b \rightarrow h$, where b is nonempty and $b \rightarrow h$ is falsified by \mathbf{x}^h . It must be that \mathbf{x}^h covers b . If \mathbf{x}^h covers b , then \mathbf{x} covers b , since no head variables may occur in b , and $\mathbf{x}^h \setminus \mathbf{x}$ consists only of those head variables besides h . Thus, changing those variables from 1 in \mathbf{x}^h to 0 in \mathbf{x} can only falsify *more* clauses. ■

The algorithm begins with an assumption that the revision distance from the initial theory to the target theory is d . If the revision fails, then d is doubled and the algorithm is repeated. Since the algorithm is later shown to be linear in d , this series of attempts does not affect the asymptotic complexity. We give a brief overview of the algorithm, followed by somewhat more detail. The pseudocode is given as Algorithm 2.

We maintain a hypothesis that is, viewed as the set of its satisfying vectors, always a superset of the target. Thus each time we ask an equivalence query, if we have not found the target, we get a negative counterexample \mathbf{x} . Then the first step is to ask a membership query on \mathbf{x} modified to turn on *all* of the head variables. If that returns 0, then the modified \mathbf{x} must falsify a headless target clause. Otherwise, for each head variable h that is 0 in the original \mathbf{x} , ask a membership query on \mathbf{x}^h . We stop when the first such membership query returns 0; we know that \mathbf{x} falsifies a clause with head h . In our pseudocode, we refer to the algorithm just described as ASSOCIATE.

Once a negative counterexample \mathbf{x} is associated with a head, we first try to use \mathbf{x} to make deletions from each existing hypothesis clause with the same head. If no such deletions are possible, then we use \mathbf{x} to add a new clause to the hypothesis. We find any necessary additions when we add a new clause.

If $(\text{body}(c) \cap \mathbf{x})^h$ (or, equivalently, $\text{body}(c)^h \cap \mathbf{x}^h$) is a negative instance, which we can determine by a membership query, then we can create a new smaller hypothesis clause whose body is $\text{body}(c) \cap \mathbf{x}$. (Notice that $\text{body}(c) \cap \mathbf{x} \subset \text{body}(c)$ because as a negative *counterexample*, \mathbf{x} must satisfy c . Furthermore, since $\text{MQ}(\mathbf{x}^{\mathbf{F}}) = 1$ and $\text{MQ}(\mathbf{x}^h) = 0$, we know that h is not in \mathbf{x} .)

To use \mathbf{x} to add a new clause, we then use an idea from the revision algorithm for monotone DNF (Goldsmith et al., 2002). For each initial theory clause with the same head as we have associated (which for \mathbf{F} is *all* initial theory clauses, since deletions of heads are allowed), use binary search from $\mathbf{x} \cap \{\text{the initial clause with the other heads set to 1}\}$ up to \mathbf{x} . If we get to something negative with fewer than d additions, we update \mathbf{x} to this negative example.

Whether or not \mathbf{x} is updated, we keep going, trying all initial theory clauses with the associated head. This guarantees that in particular we try the initial theory clause with smallest revision distance to the target clause that \mathbf{x} falsifies. All necessary additions to this clause are found by the calls to BINARYSEARCH; later only deletions will be needed.

We now give a series of lemmas that will together prove the correctness and query complexity of HORNREVISEUPTOD(ϕ, d). The first two lemmas give qualitative information. The first shows that the hypothesis is always one-sided (i.e., only negative counterexamples can ever be received), and the second says that newly added hypothesis clauses are not redundant.

Algorithm 2 HORNREVISEUPTOD(φ, d). Revises depth-1 acyclic Horn formula φ if possible using $\leq d$ revisions; otherwise returns failure.

```

1: Rewrite  $\varphi$  to remove any facts from other clauses' bodies
2:  $H :=$  everywhere-true empty conjunction
3: while ( $\mathbf{x} := \text{EQ}(H) \neq$  "Correct!" and  $d > 0$  do
4:    $h := \text{ASSOCIATE}(\mathbf{x}, \varphi)$ 
5:   if  $H$  has at least one clause then
6:     for all clauses  $c \in H$  with head  $h$  do
7:       if  $\text{MQ}((\text{body}(c) \cap \mathbf{x})^h) == 0$  then {delete vars from  $c$ }
8:          $\text{body}(c) = \text{body}(c) \cap \mathbf{x}$ 
9:          $d := d - \text{number of variables removed}$ 
10:      end if
11:    end for
12:  end if
13:  if no vars. were deleted from any clause then {find new clause to add}
14:     $\text{FoundAClause} := \text{false}; \text{min} := d$ 
15:    for all  $c \in \varphi$  with head  $h$  (or all  $c \in \varphi$  if  $h == \mathbf{F}$ ) do
16:       $\text{new} = \text{body}(c)^h \cap \mathbf{x}^h$ 
17:       $\text{numAddedVars} = 0$  {# additions to body for this  $c$ }
18:      while  $\text{MQ}(\text{new}) == 1$  and  $\text{numAddedVars} < d$  do
19:         $l := \text{BINARYSEARCH}(\mathbf{x}^h, \text{new})$ 
20:         $\text{new} := \text{new} \cup \{l\}$ 
21:         $\text{numAddedVars} := \text{numAddedVars} + 1$ 
22:        if  $\text{MQ}(\mathbf{x} - \{l\}) == 0$  then { $(\mathbf{x} - \{l\})$  is a "pivot"}
23:           $\mathbf{x} := \mathbf{x} - \{l\}$ 
24:          restart the for all  $c$  loop with this  $\mathbf{x}$ —go to Line 14 to reset other parameters
25:        end if
26:      end while
27:      if  $\text{MQ}(\text{new}) == 0$  then
28:         $\mathbf{x} := \text{new}$ 
29:         $\text{FoundAClause} := \text{true}$ 
30:         $\text{min} := \min(\text{numAddedVars}, \text{min})$ 
31:      end if
32:    end for
33:    if not  $\text{FoundAClause}$  then
34:      return "Failure"
35:    else
36:      Set all head variables of  $\mathbf{x}$  to 0
37:       $H := H \wedge (x \rightarrow h)$  {treating  $x$  as monotone disjunction}
38:       $d := d - \text{min}$ 
39:    end if
40:  end if
41: end while
42: return  $H$  is last EQ returned "Correct!", otherwise return "Failure"

```

Algorithm 3 ASSOCIATE(\mathbf{x}, φ)

```

1: if MQ( $\mathbf{x}^{\mathbf{F}}$ ) == 0 then
2:   return  $\mathbf{F}$ 
3: end if
4: for each head variable  $h$  that is 0 in  $\mathbf{x}$  do
5:   if MQ( $\mathbf{x}^h$ ) == 0 then
6:     return  $h$ 
7:   end if
8: end for

```

Lemma 1 *Algorithm HORNREVISEUPTOD maintains the invariant that its hypothesis is true for every instance that satisfies the target function.*

Proof Formally the proof is by induction on number of changes to the hypothesis after it is initialized. The base case is true, because the initial hypothesis is everywhere true.

For the inductive step, consider how we update the hypothesis, either by adding a new clause or deleting variables from the body of an existing clause.

Before creating or updating a clause to have head h and body \mathbf{y} , we have ensured (at Line 7 for updates of existing hypothesis clauses and at Line 27 for adding new clauses) that $\text{MQ}(\mathbf{y}^h) = 0$, that is, that \mathbf{y}^h is a negative instance. Because of that, \mathbf{y}^h must falsify some clause, and because of its form and the syntactic form of the target, it must be a clause with head h . None of the head variables in $\mathbf{y}^h \setminus \mathbf{y}$ can be in any body, so \mathbf{y} must indeed be a superset of the variables of some target clause with head h , as claimed. ■

Lemma 2 *Let negative counterexample \mathbf{x} be associated with head h . If \mathbf{x} is not used to make deletions, then \mathbf{x}^h falsifies any target clauses with head h whose body is covered by \mathbf{x} . Further, if \mathbf{x} is used to add a new clause with head h to the hypothesis, then the body of the new clause does not cover any target clause body covered by any other hypothesis clause with head h .*

Proof If \mathbf{x} falsified the same target clause as an existing hypothesis clause body with head h , then the membership query at Line 7 would return 0, and \mathbf{x} would be used to delete variables from that hypothesis clause body.

Now \mathbf{x} may be changed from the value it had at Line 7 before it is used to actually add a new clause. However, those changes (made when \mathbf{x} is updated at Line 28) in fact change certain non-head variables of \mathbf{x} from 1 to 0, so the updated \mathbf{x} can falsify only fewer clauses than the original \mathbf{x} . Thus if and when \mathbf{x} is used to add a new clause, \mathbf{x} cannot falsify the same target clause as any existing hypothesis clause with the same head. The newly added hypothesis clause's body is a subset of \mathbf{x} , so that clause body does not cover any other hypothesis clause body with head h . ■

The next lemma is the heart of the analysis of HORNREVISEUPTOD.

Lemma 3 *HORNREVISEUPTOD(φ, d) succeeds in finding the target Horn formula ψ if it has revision distance at most d from φ .*

Proof Let the initial formula be $\phi = \bigwedge_{i=1}^m c_i^0$ and the target formula be $\psi = \bigwedge_{i=1}^{m'} c_i^*$. We will assume throughout the analysis in this proof that the terms of the initial and target formulas are numbered so that they “line up” for calculating the revision distance from ϕ to ψ . That is, the revision distance is

$$(m - m') + \sum_{i=1}^{m'} \text{dist}(c_i^0, c_i^*),$$

where $\text{dist}(c_i^0, c_i^*)$ is the revision distance from clause c_i^0 to c_i^* , and is equal to a “body distance” that is the symmetric difference between the bodies of the two clauses, plus a “head distance” that is 1 if $\text{head}(c_i^*) = \mathbf{F}$ and $\text{head}(c_i^0) \neq \mathbf{F}$, and 0 if $\text{head}(c_i^*) = \text{head}(c_i^0)$ (and is infinite in any other case). Note that $m - m'$ accounts for the clauses deleted and that $m \geq m'$ because we cannot add entirely new clauses.

Let d_r be the value of the variable d at the start of the r th iteration of the outer **while** loop. We argue by induction on r both that d_r is an upper bound on the number of revisions required to get from a formula made of those terms in the current hypothesis and the remaining terms in the initial formula to the target, and that the r th iteration does not fail.

More precisely, assume that at the start of round r , the hypothesis H_r is $c_1 \wedge c_2 \wedge \dots \wedge c_{\ell_r}$. Part of our inductive claim is that there is a map $a(i)$ (technically a relation) of hypothesis clauses to target clauses such that

$$(\text{body}(c_i))^{\text{head}(c_i)} \text{ falsifies target clause } c_{a(i)}^*. \quad (2)$$

Formally a is a relation because some hypothesis clauses may be mapped to more than one target clause; that will occur precisely when (2) holds for more than one target clause. The relation a maps every index i of a hypothesis clause to at least one target clause index, and is one-to-one in the sense that no two target clauses ever have the same hypothesis clause mapped to both of them. The relation a evolves in only two ways: (1) when $a(i)$ is more than one index, sometimes one of those indices gets dropped, and (2) a new i gets added to the domain of a each time a clause is added to the hypothesis. For convenience of notation, we will somewhat sloppily refer to $c_{a(i)}^*$ as if it were one clause, when we mean that such statements hold for each of the associated target clauses.

The rest of the inductive claim is that: (i) the r th iteration of HORNREVISEUPTOD does not fail, and (ii) at the start of iteration r of HORNREVISEUPTOD,

$$d_r \geq \sum_{c_i \in H} \left| \text{body}(c_i) \setminus \text{body}(c_{a(i)}^*) \right| + \sum_{c_j \notin H} \text{dist}(c_j^0, c_j^*). \quad (3)$$

For the base case, $d_1 = d$, hypothesis H_1 has no terms, and Equation (3) is satisfied, since the right hand side is the revision distance from ϕ to ψ less $(m - m')$.

To complete the base case, we must argue that the first iteration does not fail. We start with a counterexample \mathbf{x} that is associated with head h . We need to show that a new clause is found using \mathbf{x} by at least one iteration of the **for all** c loop starting at Line 15. Let c_i^* be a target clause with head h or \mathbf{F} that \mathbf{x} falsifies. At some point c_i will be used as the clause in the **for all** $c \in \phi$ loop at Line 15. As long \mathbf{x}^h falsifies *only* target clause c_i^* , then after at most d calls to BINARYSEARCH, all necessary additions to c_i will have been found and a clause will be added, completing the base case. (Even if \mathbf{x}^h falsifies multiple target clauses, this still might happen.)

However, if \mathbf{x}^h falsifies more than one target clause, then we may find a variable that appears to be a necessary addition but is really a necessary addition to a different clause. Fortunately, this requires only one query to verify (see Line 22 of the algorithm). When such a variable (a “pivot”)

is found, we set that variable of \mathbf{x} to off so that the new value of \mathbf{x}^h falsifies fewer clauses. Thus, this can occur at most $m - 1$ times before \mathbf{x} falsifies exactly one clause, and no more pivots may be found. Once that happens, we must find a clause.

For the inductive step, there are two cases.

Case 1: \mathbf{x}^h is not used to delete any variables from any target clause. The argument that this iteration does not fail is the same as the corresponding argument for the base case.

As in the base case, there may be some number of times that a pivot is found and set to 0 in \mathbf{x} . Now consider the value of \mathbf{x} after any pivots have been found, and after the last time \mathbf{x} is updated at Line 27. By Lemma 2, \mathbf{x} does not cover any target clauses covered by clause bodies in the hypothesis, so it must cover one or more new target clauses. Let c_j^* be one of those target clauses. The “body” revision distance $dist(c_j^0, c_j^*)$ is equal to the number of “necessary additions”, $|\text{body}(c_j^*) \setminus \text{body}(c_j^0)|$, plus the number of “necessary deletions”, $|\text{body}(c_j^0) \setminus \text{body}(c_j^*)|$. In the iteration of the **for all** c loop at Line 15 with c set to c_j^0 , all the necessary additions had to be found, and the value of $numAddedVars$ for that iteration would have been the number of the necessary additions, so at most that number is subtracted from d_r . Also, \mathbf{x} after that intersection contained at most the variables in the body of c_j^* before the necessary deletions are made. In later revisions, all that can happen is some of those necessary deletions might happen to be made. Thus Equation (3) holds at the end of the r th iteration of the outer **while** loop. To complete the inductive step for this case, note that the relation a can indeed be extended by relating the index of the new hypothesis clause to the one or more target clauses whose body its body covers, so Equation (2) holds.

Case 2: \mathbf{x}^h is used to delete variables from at least one hypothesis clause. Say deletions are made to hypothesis clause c_i . Now $(\text{body}(c_i) \cap \mathbf{x})^h$ can falsify only the same or fewer clauses than $\text{body}(c_i)$ falsifies. By the inductive hypothesis (specifically Equation (2) coupled with Proposition 1), $\text{body}(c_i)$ falsifies target clause(s) $c_{a(i)}^*$. Thus the updated hypothesis clause $c_i := (\text{body}(c_i) \cap \mathbf{x})$ must falsify some or all of the clause(s) $c_{a(i)}^*$, and the relation a is either unchanged, or altered by decreasing the range of $a(i)$. Equation (3) still holds because we decrease d_r by the number of deletions we make, and we also decrease $|\text{body}(c_i) \setminus \text{body}(c_{a(i)}^*)|$ by the number of deletions we make.

Clause $c_{a(i)}^*$ could be derived from c_i by deletion edits; that is, $\text{body}(c_i)$ falsifies $c_{a(i)}^*$. By Proposition 1, since $\text{MQ}((\text{body}(c_i) \cap \mathbf{x})^h) = 0$, it must be that $\text{body}(c_i) \cap \mathbf{x}$ falsifies a target clause with head h . Further, using the numbering of the target clauses that makes that target clause correspond to c_i at the start of the round, the number of variables removed from $\text{body}(c)$ is subtracted from the parameter d , and Equation (3) still holds, completing the induction step.

We will find all the necessary additions to $\text{body}(c_i)$ using at most d_r calls to **BINARYSEARCH** (in fact, using at most $|\text{body}(c_i^*) \setminus \text{body}(c_i)|$) calls. Furthermore, the clause added will have all the variables in $\text{body}(c_i^*)$ and no variables not in $\text{body}(c_i) \cup \text{body}(c_i^*)$ (i.e., it will need at most only necessary deletion revisions), and the parameter d_r will be decreased by at most the number of added variables. ■

Lemma 4 *The query complexity of **HORNREVISEUPTOD**(ϕ, d) is $O(m^3 \cdot d \cdot \log n)$, where ϕ has m clauses and there are n variables in the universe.*

Proof If the variable d ever becomes nonpositive, then we terminate the algorithm.

ASSOCIATE makes at most m equivalence queries per negative counterexample. Next we try to use negative counterexample \mathbf{x} to make deletions from an existing clause. This consumes exactly 1 equivalence query and at most m membership queries. If any deletions are made, we decrease d by at least 1.

There are at most d such counterexamples used for deletions. Each counterexample used for deletions uses $\leq m + 1$ queries.

If a counterexample is not used for deletions, then we use it to add a new clause. We can have at most $m - 1$ restarts (where we back up to Line 2) due to “pivots.” These occur when \mathbf{x} falsifies multiple clauses, and each time one is found, \mathbf{x} is modified so that it falsifies fewer clauses.

There are at most m restarts, and ignoring the restarts, the main **forall** loop at Line 15 iterates over at most all m initial theory clauses. For each one iteration, the inner **while** loop iterates at most d times (once for each added literal). Each iteration of that inner **while** loop makes two direct membership queries, and one call to BINARYSEARCH, which uses at most $\log n$ queries.

Thus, each of $\leq m$ (re)starts uses at most $m \cdot d \cdot \log n$ queries, plus $2m + 1$ queries to establish that the particular counterexample should be used for the addition of a clause.

Thus the algorithm HORNREVISEUPTOD(φ, d) correctly revises initial formula φ using $O(d \cdot (m + 1) + m \cdot m^2 \cdot d \cdot \log n) = O(m^3 \cdot d \cdot \log n)$ queries. ■

Theorem 5 *There is a revision algorithm for depth-1 acyclic Horn formulas with query complexity $O(d \cdot m^3 \cdot \log n)$, where d is the revision distance, n is the number of variables in the universe, and m is the number of clauses in the initial formula.*

Proof Lemmas 3 and 4 together have shown the desired theorem. ■

3.2 An Example Run of HORNREVISEUPTOD

We now give an example run of HORNREVISEUPTOD. Suppose the variable set is $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$ and the initial formula φ and the target formula ψ are given by

$$\begin{aligned} \varphi &= x_2 \wedge (x_1 \rightarrow x_3) \wedge (x_1 \wedge x_4 \rightarrow x_5) \wedge (x_4 \wedge x_6 \rightarrow \mathbf{F}) \\ \psi &= x_2 \wedge (x_1 \rightarrow \mathbf{F}) \wedge (x_4 \wedge x_6 \wedge x_7 \wedge x_8 \rightarrow x_5). \end{aligned} \quad (4)$$

The revision distance from φ to ψ is 5: 1 for the deletion of head x_3 from second clause, 1 for the deletion of the third clause, and 3 for adding the literals \bar{x}_7 , \bar{x}_8 , and x_5 to the fourth clause (i.e., adding x_7 and x_8 to the body of the fourth clause, and x_5 to the head of the fourth clause).

For future reference, the head variables in the initial theory are x_2 , x_3 and x_5 .

Assume now that Algorithm HORNREVISEUPTOD is called with inputs φ and any $d \geq 5$. It initializes its hypothesis H to the everywhere true empty conjunction. Assume EQ(H) returns $\mathbf{x} = 11101110$, a negative counterexample. Now we call ASSOCIATE($\varphi, 11101110$) to find a candidate head for a clause negated by 11101110. In ASSOCIATE we immediately find that $\text{MQ}(11101110^{\mathbf{F}}) = \text{MQ}(11101110) = 0$, so ASSOCIATE($\varphi, 11101110$) returns \mathbf{F} . (Recall that the operation $\mathbf{x}^{\mathbf{F}}$ sets all head variables of \mathbf{x} to 1.)

Hypothesis H currently has no clauses, so we will use 11101110 to add a new clause to H starting at Line 13. Because ASSOCIATE returned \mathbf{F} , each of the four clauses of φ is considered. Say they're processed in the order they are written in Equation (4). Starting with x_2 , we set **new** to be $\text{body}(x_2)^{\mathbf{F}} \cap 11101110^{\mathbf{F}} = 01101000 \cap 11101110 = 01101000$. That is a positive instance, so we begin making calls of BINARYSEARCH from $\mathbf{x}^{\mathbf{F}} = 11101110$ to **new**. Now BINARYSEARCH(11101110,01101000) returns the position x_1 . We turn position x_1 to 1 in **new**, so now **new** is 11101000, and increment *numAddedVars* to be 1 instead of 0. Turning position x_1 to 0 in 11101110 yields a positive instance, so we do *not* have a pivot (Lines 22–25). Now $\text{MQ}(\mathbf{new}) = 0$, so we update \mathbf{x} to be 11101000, and set *FoundAClause* to true and *min* to *numAddedVars*, which is 1.

Now we have to consider the next three clauses of φ . However, when we intersect $\mathbf{x}^{\mathbf{F}} = 11101000$ with $\text{body}(c)^{\mathbf{F}}$ at Line 15 for each of the remaining three clauses c in φ we get back \mathbf{x} , so no changes are made.

Thus in Lines 36–38, we update H to be

$$H = (x_1 \rightarrow \mathbf{F}),$$

and decrement d by 1, and begin the next iteration of the outer **while** loop by making another equivalence query.

Say this time we receive the negative counterexample $\mathbf{x} = 01110111$. When we call ASSOCIATE, the instance $01110111^{\mathbf{F}} = 01111111$ is positive, so \mathbf{F} is *not* returned. The only head variable in 01110111 that is 0 is x_5 , and $\text{MQ}(01110111^{x_5}) = \text{MQ}(01110111) = 0$, so ASSOCIATE returns $h = x_5$. There is no clause in H with head x_5 , so we do not try to use instance 01110111 to delete variables from any clause of H .

In the **for** loop starting at Line 13 we consider only clauses with head x_5 ; there is exactly one: $c = (x_1 \wedge x_4 \rightarrow x_5)$. We set **new** = $\text{body}(c)^{x_5} \cap 01110111^{x_5} = 10010000^{x_5} \cap 01110111^{x_5} = 11110000 \cap 01110111 = 01110000$, which is a positive example.

Again, we make calls of BINARYSEARCH from $\mathbf{x}^h = 01110111$ to **new**. Assume that the first call returns position x_8 . Then we update **new** to 01110001, and *numAddedVars* to 1. Since **new** is still a positive instance, we call BINARYSEARCH again. Say this time it returns position x_7 . We update **new** to 01110011, and *numAddedVars* to 2. Instance **new** remains positive; we call BINARYSEARCH again; it returns x_6 ; we update **new** to 01110111 and *numAddedVars* to 3. Finally **new** is a negative instance, so we update $\mathbf{x} = \mathbf{new} = 01110111$, set *FoundAClause* to true and *min* to 3.

In Lines 36–38 we set all head variables of \mathbf{x} to 0 so $\mathbf{x} = 00010111$ and add a new clause of the form $\mathbf{x} \rightarrow h$ to H ; thus we update H to be

$$H = (x_1 \rightarrow \mathbf{F}) \wedge (x_4 \wedge x_6 \wedge x_7 \wedge x_8 \rightarrow x_5),$$

and decrement d by 3, so d has now been reduced by 4 altogether.

We begin our next iteration of the outer loop by receiving the counterexample $\mathbf{x} = 00000000$ in response to $\text{EQ}(H)$. When we call ASSOCIATE(\mathbf{x}), it determines that $00000000^{\mathbf{F}} = 01101000$ is a positive example, and so does not return \mathbf{F} , and that $00000000^{x_2} = 00101000$ is a negative example, and so does return $h = x_2$. There is no clause in H with head x_2 , so we do not try to use \mathbf{x} to delete variables from existing clauses of H .

Instead, we again execute the **for** loop starting at Line 13. This time we consider only the one clause $c = x_2$ with head x_2 (and empty body). We set $\mathbf{new} = \text{body}(x_2)^{x_2} \cap \mathbf{x}^{x_2} = 00000000^{x_2} \cap 00000000^{x_2} = 00101000$, which is a negative instance. Thus the **while** loop at Lines 18–26 is not executed at all. We skip over it and set all head variables of \mathbf{x} to 0; thus $\mathbf{x} = 00000000$. We update the hypothesis to

$$H = (x_1 \rightarrow \mathbf{F}) \wedge (x_4 \wedge x_6 \wedge x_7 \wedge x_8 \rightarrow x_5) \wedge x_2.$$

The variable *numAddedVars* was 0; so *min* was 0, and *d* is not changed from its previous value (4 less than its initial value).

Now H is the target formula, so a final equivalence query returns “Correct!” This simple example did not by any means exercise every path through the algorithm’s pseudocode, but it should give the general idea.

4. Definite Horn Formulas with Unique Heads

We give here a revision algorithm for definite Horn formulas with unique heads. A revision of a formula from class \mathcal{C} must also be in class \mathcal{C} , so in particular, a revision of a definite Horn formula also be a definite Horn formula. Thus head variables cannot be fixed to 0. We use the algorithm for revising Horn formulas in the deletions-only model presented by Goldsmith et al. (2004b) as a subroutine. Its query complexity is $O(d \cdot m^3 + m^4)$, where d is the revision distance and m is the number of clauses in the initial formula.

For this algorithm we again first give the algorithm and its analysis, and then in Section 4.2 give an example run of (the main part of) the algorithm.

Algorithm 4 DEFINITEHORNREVISE(φ). Revises φ , a definite Horn formula with unique heads

```

1:  $H :=$  everywhere-true empty conjunction
2: for all clauses  $c = (b \rightarrow h)$  of  $\varphi$  do
3:    $\mathbf{0}_h :=$  vector with 0 at  $h$ , 1’s elsewhere
4:   if  $\text{MQ}(\mathbf{0}_h) == 0$  then
5:      $\mathbf{x} :=$  vector with a 1 for every variable in  $b$  and every head of a clause of  $\varphi$  except  $h$ , and
       0’s elsewhere
6:     while  $\text{MQ}(\mathbf{x}) \neq 0$  do
7:        $v := \text{BINARYSEARCH}(\mathbf{0}_h, \mathbf{x})$ 
8:       Add variable  $v$  to clause body  $b$ 
9:       Set position  $v$  to 1 in  $\mathbf{x}$ 
10:    end while
11:    Add all heads of  $\varphi$  except  $h$  to  $b$ 
12:     $H := H \wedge (b \rightarrow h)$ 
13:   end if
14: end for
15: return DELETIONSONLYREVISE( $H$ )

```

Our algorithm, DEFINITEHORNREVISE(φ), presented as Algorithm 4, has a first phase that both deletes any clauses that need deleting in their entirety and finds all the variables that need to be added to the initial formula. That partially revised formula is then passed as an initial formula

to the known algorithm (Goldsmith et al., 2004b) for revising Horn formulas in the deletions-only model of revision.

For each clause $c = (b \rightarrow h)$, the check in Line 4 whether the vector that is 0 at h and 1 elsewhere is a negative instance determines whether clause c should be deleted altogether.

To find all necessary additions to the body b of clause $c = (b \rightarrow h)$, we use a constructed example \mathbf{x}_c . We initialize \mathbf{x}_c to b^h (the body variables from b , plus all head variables except h). Notice that the only way $\text{MQ}(\mathbf{x}_c)$ can be 0 is if \mathbf{x} covers the body of a clause but not its head. Since \mathbf{x}_c includes all heads except h , it is clear *which* clause body is or is not covered by \mathbf{x}_c ; the notion of “pivots” is not needed in this algorithm.

Next, the query $\text{MQ}(\mathbf{x}_c)$ is asked. If $\text{MQ}(\mathbf{x}_c) = 0$, then no variables need to be added to the body of c , and $b \rightarrow h$ is added to the hypothesis. If $\text{MQ}(\mathbf{x}_c) = 1$, the necessary additions to the body of c are found by repeated use of `BINARYSEARCH`. To begin the binary search, \mathbf{x}_c is the known positive instance that must satisfy the target clause c_* derived from c , and the assignment with a 0 in position h and a 1 everywhere else is the known negative instance that must falsify c_* .

Each variable returned by `BINARYSEARCH` is added to the body of the clause, and \mathbf{x}_c is updated by setting the corresponding position to 1. The process ends when \mathbf{x}_c becomes a negative instance, a clause with head h and a body variable corresponding to each 1 in $\mathbf{x}_c \rightarrow h$ is added to the hypothesis.

Once the necessary additions to every clause in the initial theory are found, a Horn formula needing only deletions has been produced, and the deletions-only algorithm `DELETIONSONLYREVISE` from (Goldsmith et al., 2004b) is used to complete the revisions.

Notice that each \mathbf{x}_c is generated, and each clause is added to the hypothesis, without any equivalence queries being asked. Thus, all additions may be made before any deletions are considered.

4.1 Analysis

The key part of the analysis of the revision complexity of this algorithm is the analysis of the initial processing of each clause. First we show that any entire clause deletions are correct, then we consider the addition of variables to an initial clause.

Lemma 6 *Algorithm `DEFINITEHORNREVISE` adds a clause that is either the initial formula clause c itself, or a revision of initial clause c made by adding variables to $\text{body}(c)$, at Line 12 if and only if some revision of c appears in the target formula.*

Proof Let $c = (b \rightarrow h)$. Vector $\mathbf{0}_h$ is 0 at position h and 1 elsewhere. If any clause that is a revision of c appears in the target (not counting the everywhere true clause, which can be omitted from any conjunction), then $\mathbf{0}_h$ must falsify this target clause. In this case, a revision of c is added to the algorithm’s hypothesis.

Conversely, if $\mathbf{0}_h$ is a positive instance, then it must be that the target contains no clause with head h , and hence, since the formulas are definite Horn formulas with unique heads, no clause that is a revision of c . In this case, the algorithm does not add any clause that is a revision of c to its hypothesis. ■

Lemma 7 *If any variable is added to the body of an initial clause c of φ in Algorithm `DEFINITEHORNREVISE`(φ), then some clause c_* that is derived from c must be in the target formula, and every variable added to c in the loop in Lines 6–9 must be in c_* .*

Proof If variables are added to the body of clause c , then eventually a clause is added to the hypothesis, and by Lemma 6, we know that this means that a clause derived from c must be in the target formula.

Variable v is added to $\text{body}(c)$ in the loop at Lines 6–9 only if there is a point in the computation where there are instances \mathbf{x} and \mathbf{x}' such \mathbf{x} is a positive instance and \mathbf{x}' is a negative instance, and \mathbf{x}' is \mathbf{x} with position v , and possibly some other positions that are not the head of any clause, changed from 0 to 1. Furthermore, \mathbf{x}' with position v set to 0 is a positive instance. By the construction, both \mathbf{x} and \mathbf{x}' must have a 1 in the position of every head except for the head h of c , so \mathbf{x}' must falsify a target clause that is a revision of c . Furthermore, since \mathbf{x}' with v set to 0 is a positive instance, v must be in that target clause. ■

From those two lemmas we can prove:

Theorem 8 *There is a revision algorithm for definite Horn formulas with unique heads in the general model of revision with query complexity $O(m^5 + d \cdot m^3 + d \cdot \log n)$, where d is the revision distance from the initial formula to the target formula, m is the number of clauses in the initial formula, and n is the number of variables in the universe.*

Proof By Lemmas 6 and 7, each variable added to a clause is necessary, and any clause deleted in the **for** loop is unnecessary.

The query complexity for the necessary additions is at most $O(\log n)$ per added variable, which contributes a factor of $O(d \log n)$.

Algorithm DELETIONS ONLY REVERSE has complexity $(m^4 + d \cdot m^3)$ (Goldsmith et al., 2004b), where m is the number of clauses in the formula to be revised, and d is the revision distance. Now the formula to given to Algorithm DELETIONS ONLY REVERSE has revision distance at most $d + m(m - 1)$, where the $m(m - 1)$ comes from the up to $m - 1$ heads added to the bodies of up to m clauses. Combining this information, we get a final query complexity of $O(m^5 + d \cdot m^3 + d \log n)$. ■

4.2 An Example Run of DEFINITEHORNREVERSE

We present an example run of DEFINITEHORNREVERSE. Suppose the variable set is $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ and the initial formula ϕ and the target formula ψ are given by

$$\begin{aligned} \phi &= x_1 \wedge (x_5 \wedge x_6 \rightarrow x_2) \wedge (x_2 \wedge x_3 \wedge x_4 \rightarrow x_5) \wedge (x_2 \rightarrow x_6) \\ \psi &= (x_3 \rightarrow x_1) \wedge (x_5 \wedge x_6 \rightarrow x_2) \wedge (x_3 \wedge x_4 \wedge x_6 \rightarrow x_5) \end{aligned} \quad (5)$$

The revision distance from ϕ to ψ is 4: 1 for adding x_3 to the body of the first clause, 2 for adding one variable and deleting another from the body of the third clause, and 1 for deleting the fourth clause.

We process the four clauses of ϕ in order:

1. *Clause x_1* : The vector $\mathbf{0}_{x_1} = 011111$ (i.e., 0 only at x_1) is a negative instance, so we retain this clause. We set \mathbf{x} to 010011, which is a positive instance, so we enter the **while** loop of Algorithm DEFINITEHORNREVERSE at Lines 6–10. In the first iteration BINARYSEARCH($\mathbf{0}_{x_1}, \mathbf{x}$)

returns x_3 . Setting x_3 to 1 in \mathbf{x} makes $\mathbf{x} = 011011$, which is a negative instance, so we are done with the **while** loop.

We insert the clause

$$(x_2 \wedge x_3 \wedge x_5 \wedge x_6 \rightarrow x_1)$$

into the hypothesis.

2. *Clause* $(x_5 \wedge x_6 \rightarrow x_2)$: Vector $\mathbf{0}_{x_2} = 101111$ is negative, so we retain this clause. We set $\mathbf{x} = 100011$, which is a negative instance, so we do not need to call **BINARYSEARCH**.

We insert the clause

$$(x_1 \wedge x_5 \wedge x_6 \rightarrow x_2)$$

into the hypothesis.

3. *Clause* $(x_2 \wedge x_3 \wedge x_4 \rightarrow x_5)$: Vector $\mathbf{0}_{x_5} = 111101$ is negative, so we retain this clause. We set $\mathbf{x} = 111101$, which is a negative instance, so we do not need to call **BINARYSEARCH**.

We insert the clause

$$(x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_6 \rightarrow x_5)$$

into the hypothesis.

4. *Clause* $(x_2 \rightarrow x_6)$: Vector $\mathbf{0}_{x_6} = 111110$ is positive, so we do not put this clause in the hypothesis (i.e., we delete this clause).

Our hypothesis is now

$$H = (x_2 \wedge x_3 \wedge x_5 \wedge x_6 \rightarrow x_1) \wedge (x_1 \wedge x_5 \wedge x_6 \rightarrow x_2) \wedge (x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_6 \rightarrow x_5).$$

We now have a hypothesis needing only deletion edits, and we pass this hypothesis to the deletions-only algorithm from Goldsmith et al. (2004b).

5. Conclusions and Open Questions

Horn formulas are ubiquitous in Computer Science, occurring in subfields from expert systems to databases. In all of these instances, the formulas or theories are dependent on human expertise or on potentially changing conditions. In many cases, “oracles” capable of answering equivalence or membership queries are far easier to come by than are direct sources for the correct theories.

The problem of revising Horn formulas with queries remains open, but this paper has broken the additions barrier. Open questions range from small to large improvements on the results presented here. For instance, we have presented a revision algorithm for acyclic, depth-1 Horn formulas. Can the techniques used here be extended to acyclic depth-2 or depth- k formulas? For acyclic formulas with unbounded depth? How much harder is it to revise Horn formulas with unique heads if we allow up to one occurrence of \mathbf{F} as a head? Bounded or unbounded occurrences?

If we can revise acyclic depth- k Horn formulas for each k , how does the complexity depend on k ? Is the problem fixed-parameter tractable? Could the complexity of revision depend on a fixed maximum number of occurrences as the head of a clause for each variable?

Acknowledgments

This work was partially supported by NSF grants CCR-0100040 and ITR-0325063 to the first author, and NSF grants CCR-0100336 and CCF-0431059 to the second author. The authors also thank Jignesh Doshi for inspiring this work, and Balázs Szörényi and György Turán for publishing their results with a preliminary version of the results given here, in a joint ALT'04 paper (Goldsmith et al., 2004a).

References

- Dana Angluin. Learning propositional Horn sentences with hints. Technical Report YALEU/DCS/RR-590, Department of Computer Science, Yale University, December 1987a.
- Dana Angluin. Learning regular sets from queries and counterexamples. *Inform. Comput.*, 75(2): 87–106, November 1987b.
- Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.
- Dana Angluin, Michael Frazier, and Leonard Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.
- Krzysztof R. Apt, Howard Blair, and Adrian Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, Los Altos, CA, 1988.
- Hiroki Arimura. Learning acyclic first-order Horn sentences from entailment. In *Algorithmic Learning Theory, 8th International Workshop, ALT '97, Sendai, Japan, October 1997, Proceedings*, volume 1316 of *Lecture Notes in Artificial Intelligence*, pages 432–445. Springer, 1997.
- Peter Auer and Philip M. Long. Structural results about on-line learning models with and without queries. *Machine Learning*, 36(3):147–181, 1999.
- Avrim Blum, Lisa Hellerstein, and Nick Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *J. of Comput. Syst. Sci.*, 50(1):32–40, 1995. Earlier version in 4th COLT, 1991.
- Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5:649–667, 2004.
- Nader Bshouty. Exact learning Boolean function via the monotone theory. *Information and Computation*, 123:146–153, 1995.
- Nader Bshouty and Lisa Hellerstein. Attribute-efficient learning in query and mistake-bound models. *J. of Comput. Syst. Sci.*, 56(3):310–319, 1998.
- Ashok K. Chandra and David Harel. Horn clause queries and generalizations. *Journal of Logic Programming*, 2:1–15, 1985.
- Jignesh Umesh Doshi. Revising Horn formulas. Master's thesis, Dept. of Computer Science, University of Kentucky, 2003.

- Judy Goldsmith, Robert H. Sloan, and György Turán. Theory revision with queries: DNF formulas. *Machine Learning*, 47(2/3):257–295, 2002.
- Judy Goldsmith, Robert H. Sloan, Balázs Szörényi, and György Turán. New revision algorithms. In *Algorithmic Learning Theory, 15th International Conference, ALT 2004, Padova, Italy, October 2004, Proceedings*, volume 3244 of *Lecture Notes in Artificial Intelligence*, pages 395–409. Springer, 2004a.
- Judy Goldsmith, Robert H. Sloan, Balázs Szörényi, and György Turán. Theory revision with queries: Horn, read-once, and parity formulas. *Artificial Intelligence*, 156:139–176, 2004b.
- Russell Greiner. The complexity of theory revision. *Artificial Intelligence*, 107:175–217, 1999a.
- Russell Greiner. The complexity of revising logic programs. *J. Logic Programming*, 40:273–298, 1999b.
- Peter L. Hammer and Alexander Kogan. Quasi-acyclic propositional Horn knowledge bases: optimal compression. *IEEE Trans. Knowl. Data Eng.*, 7:751–762, 1995.
- Moshe Koppel, Ronen Feldman, and Alberto Maria Segre. Bias-driven revision of logical domain theories. *Journal of Artificial Intelligence Research*, 1:159–208, 1994.
- Evelina Lamma, Fabrizio Riguzzi, and Luís Moniz Pereira. Belief revision via Lamarckian evolution. *New Generation Computing*, 21:247–275, 2003.
- Raymond J. Mooney. A preliminary PAC analysis of theory revision. In Thomas Petsche, editor, *Computational Learning Theory and Natural Learning Systems*, volume III: Selecting Good Models, chapter 3, pages 43–53. MIT Press, 1995.
- Dirk Ourston and Raymond J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:273–309, 1994.
- Bradley L. Richards and Raymond J. Mooney. Automated refinement of first-order Horn-clause domain theories. *Machine Learning*, 19:95–131, 1995.
- Robert H. Sloan, Balázs Szörényi, and György Turán. Projective DNF formulae and their revision. In *Learning Theory and Kernel Machines, 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 625–639. Springer, 2003.
- Geoffrey G. Towell and Jude W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.
- Allen Van Gelder. Negation as failure using tight derivations for general logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 149–176. Morgan Kaufmann, Los Altos, CA, 1988.
- Stefan Wrobel. First order theory refinement. In L. De Raedt, editor, *Advances in ILP*, pages 14–33. IOS Press, Amsterdam, 1995.
- Stefan Wrobel. *Concept Formation and Knowledge Revision*. Kluwer, 1994.

A Unifying View of Sparse Approximate Gaussian Process Regression

Joaquin Quiñero-Candela

Carl Edward Rasmussen

Max Planck Institute for Biological Cybernetics

Spemannstraße 38

72076 Tübingen, Germany

JQC@TUEBINGEN.MPG.DE

CARL@TUEBINGEN.MPG.DE

Editor: Ralf Herbrich

Abstract

We provide a new unifying view, including all existing proper probabilistic sparse approximations for Gaussian process regression. Our approach relies on expressing the *effective prior* which the methods are using. This allows new insights to be gained, and highlights the relationship between existing methods. It also allows for a clear theoretically justified ranking of the closeness of the known approximations to the corresponding full GPs. Finally we point directly to designs of new better sparse approximations, combining the best of the existing strategies, within attractive computational constraints.

Keywords: Gaussian process, probabilistic regression, sparse approximation, Bayesian committee machine

Regression models based on Gaussian processes (GPs) are simple to implement, flexible, fully probabilistic models, and thus a powerful tool in many areas of application. Their main limitation is that memory requirements and computational demands grow as the square and cube respectively, of the number of training cases n , effectively limiting a direct implementation to problems with at most a few thousand cases. To overcome the computational limitations numerous authors have recently suggested a wealth of *sparse* approximations. Common to all these approximation schemes is that only a subset of the latent variables are treated exactly, and the remaining variables are given some approximate, but computationally cheaper treatment. However, the published algorithms have widely different motivations, emphasis and exposition, so it is difficult to get an overview (see Rasmussen and Williams, 2006, chapter 8) of how they relate to each other, and which can be expected to give rise to the best algorithms.

In this paper we provide a unifying view of sparse approximations for GP regression. Our approach is simple, but powerful: for each algorithm we analyze the posterior, and compute the *effective prior* which it is using. Thus, we reinterpret the algorithms as “exact inference with an approximated prior”, rather than the existing (ubiquitous) interpretation “approximate inference with the exact prior”. This approach has the advantage of directly expressing the approximations in terms of prior assumptions about the function, which makes the consequences of the approximations much easier to understand. While our view of the approximations is not the only one possible, it has the advantage of putting all existing probabilistic sparse approximations under one umbrella, thus enabling direct comparison and revealing the relation between them.

In Section 1 we briefly introduce GP models for regression. In Section 2 we present our unifying framework and write out the key equations in preparation for the unifying analysis of sparse

algorithms in Sections 4-7. The relation of transduction and augmentation to our sparse framework is covered in Section 8. All our approximations are written in terms of a new set of *inducing variables*. The choice of these variables is itself a challenging problem, and is discussed in Section 9. We comment on a few special approximations outside our general scheme in Section 10 and conclusions are drawn at the end.

1. Gaussian Processes for Regression

Probabilistic regression is usually formulated as follows: given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$ of n pairs of (vectorial) inputs \mathbf{x}_i and noisy (real, scalar) outputs y_i , compute the predictive distribution of the function values f_* (or noisy y_*) at test locations \mathbf{x}_* . In the simplest case (which we deal with here) we assume that the noise is additive, independent and Gaussian, such that the relationship between the (latent) function $f(\mathbf{x})$ and the observed noisy targets y are given by

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \quad \text{where } \varepsilon_i \sim \mathcal{N}(0, \sigma_{\text{noise}}^2), \quad (1)$$

where σ_{noise}^2 is the variance of the noise.

Definition 1 *A Gaussian process (GP) is a collection of random variables, any finite number of which have consistent¹ joint Gaussian distributions.*

Gaussian process (GP) regression is a Bayesian approach which assumes a GP prior² over functions, i.e. assumes a priori that function values behave according to

$$p(\mathbf{f} | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = \mathcal{N}(\mathbf{0}, K), \quad (2)$$

where $\mathbf{f} = [f_1, f_2, \dots, f_n]^\top$ is a vector of latent function values, $f_i = f(\mathbf{x}_i)$ and K is a covariance matrix, whose entries are given by the *covariance function*, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Note that the GP treats the latent function values f_i as random variables, indexed by the corresponding input. In the following, for simplicity we will always neglect the explicit conditioning on the inputs; the GP model and all expressions are always conditional on the corresponding inputs. The GP model is concerned only with the conditional of the outputs given the inputs; we do not model anything about the inputs themselves.

Remark 2 *Note, that to adhere to a strict Bayesian formalism, the GP covariance function,³ which defines the prior, should not depend on the data (although it can depend on additional parameters).*

As we will see in later sections, some approximations are strictly equivalent to GPs, while others are not. That is, the implied prior may still be multivariate Gaussian, but the covariance function may be different for training and test cases.

Definition 3 *A Gaussian process is called degenerate iff the covariance function has a finite number of non-zero eigenvalues.*

1. By consistency is meant simply that the random variables obey the usual rules of marginalization, etc.
 2. For notational simplicity we exclusively use zero-mean priors.
 3. The covariance *function* itself shouldn't depend on the data, though its value at a specific pair of inputs of course will.

Degenerate GPs (such as e.g. with polynomial covariance function) correspond to *finite* linear (-in-the-parameters) models, whereas non-degenerate GPs (such as e.g. with squared exponential or RBF covariance function) do not. The prior for a finite m dimensional linear model only considers a universe of at most m linearly independent functions; this may often be too restrictive when $n \gg m$. Note however, that non-degeneracy on its own doesn't guarantee the existence of the "right kind" of flexibility for a given particular modelling task. For a more detailed background on GP models, see for example that of Rasmussen and Williams (2006).

Inference in the GP model is simple: we put a joint GP prior on training and test latent values, \mathbf{f} and \mathbf{f}_* ⁴, and combine it with the likelihood⁵ $p(\mathbf{y}|\mathbf{f})$ using Bayes rule, to obtain the joint posterior

$$p(\mathbf{f}, \mathbf{f}_* | \mathbf{y}) = \frac{p(\mathbf{f}, \mathbf{f}_*)p(\mathbf{y}|\mathbf{f})}{p(\mathbf{y})}. \quad (3)$$

The final step needed to produce the desired posterior predictive distribution is to marginalize out the unwanted training set latent variables:

$$p(\mathbf{f}_* | \mathbf{y}) = \int p(\mathbf{f}, \mathbf{f}_* | \mathbf{y}) d\mathbf{f} = \frac{1}{p(\mathbf{y})} \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, \mathbf{f}_*) d\mathbf{f}, \quad (4)$$

or in words: the predictive distribution is the marginal of the renormalized joint prior times the likelihood. The joint GP prior and the independent likelihood are both Gaussian

$$p(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{\mathbf{f},\mathbf{f}} & K_{*,\mathbf{f}} \\ K_{\mathbf{f},*} & K_{*,*} \end{bmatrix}\right), \quad \text{and} \quad p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_{\text{noise}}^2 I), \quad (5)$$

where K is subscript by the variables between which the covariance is computed (and we use the asterisk $*$ as shorthand for \mathbf{f}_*) and I is the identity matrix. Since both factors in the integral are Gaussian, the integral can be evaluated in closed form to give the Gaussian predictive distribution

$$p(\mathbf{f}_* | \mathbf{y}) = \mathcal{N}\left(K_{*,\mathbf{f}}(K_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I)^{-1}\mathbf{y}, K_{*,*} - K_{*,\mathbf{f}}(K_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I)^{-1}K_{\mathbf{f},*}\right), \quad (6)$$

see the relevant Gaussian identity in appendix A. The problem with the above expression is that it requires inversion of a matrix of size $n \times n$ which requires $O(n^3)$ operations, where n is the number of training cases. Thus, the simple exact implementation can handle problems with at most a few thousand training cases.

2. A New Unifying View

We now seek to modify the joint prior $p(\mathbf{f}_*, \mathbf{f})$ from (5) in ways which will reduce the computational requirements from (6). Let us first rewrite that prior by introducing an additional set of m latent variables $\mathbf{u} = [u_1, \dots, u_m]^\top$, which we call the *inducing variables*. These latent variables are values of the Gaussian process (as also \mathbf{f} and \mathbf{f}_*), corresponding to a set of input locations $X_{\mathbf{u}}$, which we call the *inducing inputs*. Whereas the additional latent variables \mathbf{u} are always marginalized out in the predictive distribution, the choice of inducing inputs *does* leave an imprint on the final solution.

4. We will mostly consider a vector of test cases \mathbf{f}_* (rather than a single f_*).

5. You may have been expecting the likelihood written as $p(\mathbf{y}|\mathbf{f}_*, \mathbf{f})$ but since the likelihood is conditionally independent of everything else given \mathbf{f} , this makes no difference.

The inducing variables will turn out to be generalizations of variables which other authors have referred to variously as “support points”, “active set” or “pseudo-inputs”. Particular sparse algorithms choose the inducing variables in various different ways; some algorithms chose the inducing inputs to be a subset of the training set, others not, as we will discuss in Section 9. For now consider any arbitrary inducing variables.

Due to the *consistency* of Gaussian processes, we know that we can recover $p(\mathbf{f}_*, \mathbf{f})$ by simply integrating (marginalizing) out \mathbf{u} from the joint GP prior $p(\mathbf{f}_*, \mathbf{f}, \mathbf{u})$

$$p(\mathbf{f}_*, \mathbf{f}) = \int p(\mathbf{f}_*, \mathbf{f}, \mathbf{u}) \, d\mathbf{u} = \int p(\mathbf{f}_*, \mathbf{f}|\mathbf{u}) p(\mathbf{u}) \, d\mathbf{u}, \quad \text{where } p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{u},\mathbf{u}}). \quad (7)$$

This is an exact expression. Now, we introduce the fundamental approximation which gives rise to almost all sparse approximations. We approximate the joint prior by assuming that \mathbf{f}_* and \mathbf{f} are *conditionally independent given \mathbf{u}* , see Figure 1, such that

$$p(\mathbf{f}_*, \mathbf{f}) \simeq q(\mathbf{f}_*, \mathbf{f}) = \int q(\mathbf{f}_*|\mathbf{u}) q(\mathbf{f}|\mathbf{u}) p(\mathbf{u}) \, d\mathbf{u}. \quad (8)$$

The name *inducing* variable is motivated by the fact that \mathbf{f} and \mathbf{f}_* can only communicate through \mathbf{u} , and \mathbf{u} therefore *induces* the dependencies between training and test cases. As we shall detail in the following sections, the different computationally efficient algorithms proposed in the literature correspond to different *additional assumptions* about the two approximate *inducing* conditionals $q(\mathbf{f}|\mathbf{u})$, $q(\mathbf{f}_*|\mathbf{u})$ of the integral in (8). It will be useful for future reference to specify here the exact expressions for the two conditionals

$$\text{training conditional: } p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}), \quad (9a)$$

$$\text{test conditional: } p(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(K_{*,\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, K_{*,*} - Q_{*,*}), \quad (9b)$$

where we have introduced the shorthand notation⁶ $Q_{\mathbf{a},\mathbf{b}} \triangleq K_{\mathbf{a},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} K_{\mathbf{u},\mathbf{b}}$. We can readily identify the expressions in (9) as special (noise free) cases of the standard predictive equation (6) with \mathbf{u} playing the role of (noise free) observations. Note that the (positive semi-definite) covariance matrices in (9) have the form $K - Q$ with the following interpretation: the prior covariance K minus a (non-negative definite) matrix Q quantifying how much information \mathbf{u} provides about the variables in question (\mathbf{f} or \mathbf{f}_*). We emphasize that all the sparse methods discussed in the paper correspond simply to different approximations to the conditionals in (9), and throughout we use the exact likelihood and inducing prior

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_{\text{noise}}^2 I), \quad \text{and } p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{u},\mathbf{u}}). \quad (10)$$

3. The Subset of Data (SoD) Approximation

Before we get started with the more sophisticated approximations, we mention as a baseline method the simplest possible sparse approximation (which doesn’t fall inside our general scheme): use only a subset of the data (SoD). The computational complexity is reduced to $O(m^3)$, where $m < n$. We would not generally expect SoD to be a competitive method, since it would seem impossible (even with fairly redundant data and a good choice of the subset) to get a realistic picture of the

6. Note, that $Q_{\mathbf{a},\mathbf{b}}$ depends on \mathbf{u} although this is not explicit in the notation.

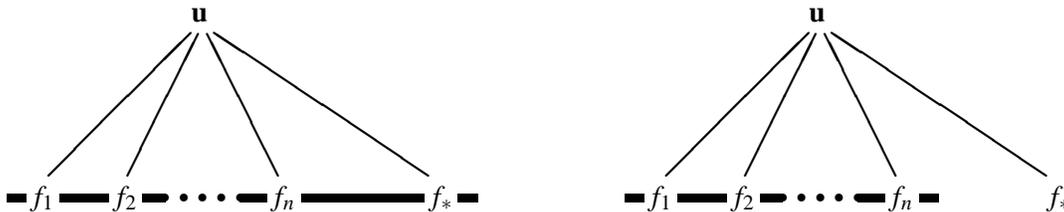


Figure 1: Graphical model of the relation between the inducing variables \mathbf{u} , the training latent functions values $\mathbf{f} = [f_1, \dots, f_n]^\top$ and the test function value f_* . The thick horizontal line represents a set of fully connected nodes. The observations y_1, \dots, y_n, y_* (not shown) would dangle individually from the corresponding latent values, by way of the exact (factored) likelihood (5). **Left graph:** the fully connected graph corresponds to the case where no approximation is made to the full joint Gaussian process distribution between these variables. The inducing variables \mathbf{u} are superfluous in this case, since all latent function values can communicate with all others. **Right graph:** assumption of *conditional independence* between training and test function values given \mathbf{u} . This gives rise to the separation between training and test conditionals from (8). Notice that having cut the communication path between training and test latent function values, information from \mathbf{f} can only be transmitted to f_* via the inducing variables \mathbf{u} .

uncertainties, when only a part of the training data is even considered. We include it here mostly as a baseline against which to compare better sparse approximations.

In Figure 5 top, left we see how the SoD method produces wide predictive distributions, when training on a randomly selected subset of 10 cases. A fair comparison to other methods would take into account that the computational complexity is independent of n as opposed to other more advanced methods. These extra computational resources could be spent in a number of ways, e.g. larger m , or an active (rather than random) selection of the m points. In this paper we will concentrate on understanding the theoretical foundations of the various approximations rather than investigating the necessary heuristics needed to turn the approximation schemes into actually practical algorithms.

4. The Subset of Regressors (SoR) Approximation

The Subset of Regressors (SoR) algorithm was given by Silverman (1985), and mentioned again by Wahba et al. (1999). It was then adapted by Smola and Bartlett (2001) to propose a sparse greedy approximation to Gaussian process regression. SoR models are finite linear-in-the-parameters models with a particular prior on the weights. For any input \mathbf{x}_* , the corresponding function value f_* is given by:

$$f_* = K_{*,\mathbf{u}} \mathbf{w}_{\mathbf{u}}, \quad \text{with} \quad p(\mathbf{w}_{\mathbf{u}}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{u},\mathbf{u}}^{-1}), \quad (11)$$

where there is one weight associated to each inducing input in $X_{\mathbf{u}}$. Note that the covariance matrix for the prior on the weights is the *inverse* of that on \mathbf{u} , such that we recover the exact GP prior on \mathbf{u} ,

which is Gaussian with zero mean and covariance

$$\mathbf{u} = K_{\mathbf{u},\mathbf{u}} \mathbf{w}_{\mathbf{u}} \Rightarrow \langle \mathbf{u} \mathbf{u}^\top \rangle = K_{\mathbf{u},\mathbf{u}} \langle \mathbf{w}_{\mathbf{u}} \mathbf{w}_{\mathbf{u}}^\top \rangle K_{\mathbf{u},\mathbf{u}} = K_{\mathbf{u},\mathbf{u}}. \quad (12)$$

Using the effective prior on \mathbf{u} and the fact that $\mathbf{w}_{\mathbf{u}} = K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}$ we can redefine the SoR model in an equivalent, more intuitive way:

$$\mathbf{f}_* = K_{*,\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, \quad \text{with } \mathbf{u} \sim \mathcal{N}(\mathbf{0}, K_{\mathbf{u},\mathbf{u}}). \quad (13)$$

We are now ready to integrate the SoR model in our unifying framework. Given that there is a *deterministic* relation between any \mathbf{f}_* and \mathbf{u} , the approximate conditional distributions in the integral in eq. (8) are given by:

$$q_{\text{SoR}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, \mathbf{0}), \quad \text{and} \quad q_{\text{SoR}}(\mathbf{f}_*|\mathbf{u}) = \mathcal{N}(K_{*,\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, \mathbf{0}), \quad (14)$$

with zero conditional covariance, compare to (9). The effective prior implied by the SoR approximation is easily obtained from (8), giving

$$q_{\text{SoR}}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & Q_{*,*} \end{bmatrix}\right), \quad (15)$$

where we recall $Q_{\mathbf{a},\mathbf{b}} \triangleq K_{\mathbf{a},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} K_{\mathbf{u},\mathbf{b}}$. A more descriptive name for this method, would be the Deterministic Inducing Conditional (DIC) approximation. We see that this approximate prior is degenerate. There are only m degrees of freedom in the model, which implies that only m linearly independent functions can be drawn from the prior. The $m+1$ -th one is a linear combination of the previous. For example, in a very low noise regime, the posterior could be severely constrained by only m training cases.

The degeneracy of the prior causes unreasonable predictive distributions. Indeed, the approximate prior over functions is so restrictive, that given enough data only a very limited family of functions will be plausible under the posterior, leading to overconfident predictive variances. This is a general problem of finite linear models with small numbers of weights (for more details see Rasmussen and Quiñonero-Candela, 2005). Figure 5, top, right panel, illustrates the unreasonable predictive uncertainties of the SoR approximation on a toy dataset.⁷

The predictive distribution is obtained by using the SoR approximate prior (15) instead of the true prior in (4). For each algorithm we give two forms of the predictive distribution, one which is easy to interpret, and the other which is economical to compute with:

$$q_{\text{SoR}}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I)^{-1} \mathbf{y}, Q_{*,*} - Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I)^{-1} Q_{\mathbf{f},*}), \quad (16a)$$

$$= \mathcal{N}(\sigma^{-2} K_{*,\mathbf{u}} \Sigma K_{\mathbf{u},\mathbf{f}} \mathbf{y}, K_{*,\mathbf{u}} \Sigma K_{\mathbf{u},*}), \quad (16b)$$

where we have defined $\Sigma = (\sigma^{-2} K_{\mathbf{u},\mathbf{f}} K_{\mathbf{f},\mathbf{u}} + K_{\mathbf{u},\mathbf{u}})^{-1}$. Equation (16a) is readily recognized as the regular prediction equation (6), except that the covariance K has everywhere been replaced by Q , which was already suggested by (15). This corresponds to replacing the covariance function k with $k_{\text{SoR}}(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{u}) K_{\mathbf{u},\mathbf{u}}^{-1} k(\mathbf{u}, \mathbf{x}_j)$. The new covariance function has rank (at most) m . Thus we have the following

7. Wary of this fact, Smola and Bartlett (2001) propose using the predictive variances of the SoD, or a more accurate computationally costly alternative (more details are given by Quiñonero-Candela, 2004, Chapter 3).

Remark 4 *The SoR approximation is equivalent to exact inference in the degenerate Gaussian process with covariance function $k_{\text{SoR}}(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{u})K_{\mathbf{u},\mathbf{u}}^{-1}k(\mathbf{u}, \mathbf{x}_j)$.*

The equivalent (16b) is computationally cheaper, and with (11) in mind, Σ is the covariance of the posterior on the weights $\mathbf{w}_{\mathbf{u}}$. Note that as opposed to the subset of data method, all training cases are taken into account. The computational complexity is $O(nm^2)$ initially, and $O(m)$ and $O(m^2)$ per test case for the predictive mean and variance respectively.

5. The Deterministic Training Conditional (DTC) Approximation

Taking up ideas already contained in the work of Csató and Opper (2002), Seeger et al. (2003) recently proposed another sparse approximation to Gaussian process regression, which does not suffer from the nonsensical predictive uncertainties of the SoR approximation, but that interestingly leads to exactly the same predictive mean. Seeger et al. (2003), who called the method Projected Latent Variables (PLV), presented the method as relying on a *likelihood* approximation, based on the projection $\mathbf{f} = K_{\mathbf{f},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}$:

$$p(\mathbf{y}|\mathbf{f}) \simeq q(\mathbf{y}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \sigma_{\text{noise}}^2 I). \quad (17)$$

The method has also been called the Projected Process Approximation (PPA) by Rasmussen and Williams (2006, Chapter 8). One way of obtaining an equivalent model is to retain the usual likelihood, but to impose a deterministic training conditional and the exact test conditional from eq. (9b)

$$q_{\text{DTC}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \mathbf{0}), \quad \text{and} \quad q_{\text{DTC}}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u}). \quad (18)$$

This reformulation has the advantage of allowing us to stick to our view of exact inference (with exact likelihood) with approximate priors. Indeed, under this model the conditional distribution of \mathbf{f} given \mathbf{u} is identical to that of the SoR, given in the left of (14). A systematic name for this approximation is the Deterministic Training Conditional (DTC).

The fundamental difference with SoR is that DTC uses the exact test conditional (9b) instead of the deterministic relation between \mathbf{f}_* and \mathbf{u} of SoR. The joint prior implied by DTC is given by:

$$q_{\text{DTC}}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}\right), \quad (19)$$

which is surprisingly similar to the effective prior implied by the SoR approximation (15). The fundamental difference is that under the DTC approximation \mathbf{f}_* has a prior variance of its own, given by $K_{*,*}$. This prior variance reverses the behaviour of the predictive uncertainties, and turns them into sensible ones, see Figure 5 for an illustration.

The predictive distribution is now given by:

$$q_{\text{DTC}}(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I)^{-1}\mathbf{y}, K_{*,*} - Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I)^{-1}Q_{\mathbf{f},*}) \quad (20a)$$

$$= \mathcal{N}(\sigma^{-2}K_{*,\mathbf{u}}\Sigma K_{\mathbf{u},\mathbf{f}}\mathbf{y}, K_{*,*} - Q_{*,*} + K_{*,\mathbf{u}}\Sigma K_{*,\mathbf{u}}^\top), \quad (20b)$$

where again we have defined $\Sigma = (\sigma^{-2}K_{\mathbf{u},\mathbf{f}}K_{\mathbf{f},\mathbf{u}} + K_{\mathbf{u},\mathbf{u}})^{-1}$ as in (16). The predictive mean for the DTC is identical to that of the SoR approximation (16), but the predictive variance replaces the $Q_{*,*}$ from SoR with $K_{*,*}$ (which is larger, since $K_{*,*} - Q_{*,*}$ is positive definite). This added term is the predictive variance of the posterior of f_* conditioned on \mathbf{u} . It grows to the prior variance $K_{*,*}$ as \mathbf{x}_* moves far from the inducing inputs in $X_{\mathbf{u}}$.

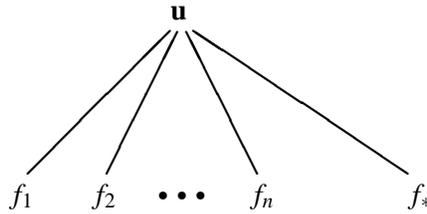


Figure 2: Graphical model for the FITC approximation. Compared to those in Figure 1, all edges between latent function values have been removed: the latent function values are conditionally fully independent given the inducing variables \mathbf{u} . Although strictly speaking the SoR and DTC approximations could also be represented by this graph, note that both further assume a deterministic relation between \mathbf{f} and \mathbf{u} .

Remark 5 *The only difference between the predictive distribution of DTC and SoR is the variance. The predictive variance of DTC is never smaller than that of SoR.*

Note, that since the covariances for training cases and test cases are computed differently, see (19), it follows that

Remark 6 *The DTC approximation does not correspond exactly to a Gaussian process,*

as the covariance between latent values depends on whether they are considered training or test cases, violating consistency, see Definition 1. The computational complexity has the same order as for SoR.

6. The Fully Independent Training Conditional (FITC) Approximation

Recently Snelson and Ghahramani (2006) proposed another likelihood approximation to speed up Gaussian process regression, which they called Sparse Gaussian Processes using Pseudo-inputs (SGPP). While the DTC is based on the likelihood approximation given by (17), the SGPP proposes a more sophisticated likelihood approximation with a richer covariance

$$p(\mathbf{y}|\mathbf{f}) \simeq q(\mathbf{y}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, \text{diag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}] + \sigma_{\text{noise}}^2 I), \quad (21)$$

where $\text{diag}[A]$ is a diagonal matrix whose elements match the diagonal of A . As we did in (18) for the DTC, we provide an alternative equivalent formulation called Fully Independent Training Conditional (FITC) based on the inducing conditionals:

$$q_{\text{FITC}}(\mathbf{f}|\mathbf{u}) = \prod_{i=1}^n p(f_i|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}} K_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}, \text{diag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}]), \quad \text{and} \quad q_{\text{FITC}}(f_*|\mathbf{u}) = p(f_*|\mathbf{u}). \quad (22)$$

We see that as opposed to SoR and DTC, FITC does not impose a deterministic relation between \mathbf{f} and \mathbf{u} . Instead of ignoring the variance, FITC proposes an approximation to the training conditional distribution of \mathbf{f} given \mathbf{u} as a further independence assumption. In addition, the exact test conditional from (9b) is used in (22), although for reasons which will become clear towards the end of this

section, we initially consider only a single test case, f_* . The corresponding graphical model is given in Figure 2. The effective prior implied by the FITC is given by

$$q_{\text{FITC}}(\mathbf{f}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \text{diag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}\right). \quad (23)$$

Note, that the sole difference between the DTC and FITC is that in the top left corner of the implied prior covariance, FITC replaces the approximate covariances of DTC by the exact ones on the diagonal. The predictive distribution is

$$q_{\text{FITC}}(f_*|\mathbf{y}) = \mathcal{N}(Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \Lambda)^{-1}\mathbf{y}, K_{*,*} - Q_{*,\mathbf{f}}(Q_{\mathbf{f},\mathbf{f}} + \Lambda)^{-1}Q_{\mathbf{f},*}) \quad (24a)$$

$$= \mathcal{N}(K_{*,\mathbf{u}}\Sigma K_{\mathbf{u},\mathbf{f}}\Lambda^{-1}\mathbf{y}, K_{*,*} - Q_{*,*} + K_{*,\mathbf{u}}\Sigma K_{\mathbf{u},*}), \quad (24b)$$

where we have defined $\Sigma = (K_{\mathbf{u},\mathbf{u}} + K_{\mathbf{u},\mathbf{f}}\Lambda^{-1}K_{\mathbf{f},\mathbf{u}})^{-1}$ and $\Lambda = \text{diag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2\mathbf{I}]$. The computational complexity is identical to that of SoR and DTC.

So far we have only considered a single test case. There are two options for joint predictions, either 1) use the exact full test conditional from (9b), or 2) extend the additional factorizing assumption to the test conditional. Although Snelson and Ghahramani (2006) don't explicitly discuss joint predictions, it would seem that they probably intend the second option. Whereas the additional independence assumption for the test cases is not really necessary for computational reasons, it does affect the nature of the approximation. Under option 1) the training and test covariance are computed differently, and thus this does not correspond to our strict definition of a GP model, but

Remark 7 *If the assumption of full independence is extended to the test conditional, the FITC approximation is equivalent to exact inference in a non-degenerate Gaussian process with covariance function $k_{\text{FIC}}(x_i, x_j) = k_{\text{SoR}}(x_i, x_j) + \delta_{i,j}[k(x_i, x_j) - k_{\text{SoR}}(x_i, x_j)]$,*

where $\delta_{i,j}$ is Kronecker's delta. A logical name for the method where the conditionals (training and test) are always forced to be fully independent would be the Fully Independent Conditional (FIC) approximation. The effective prior implied by FIC is:

$$q_{\text{FIC}}(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \text{diag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & Q_{*,*} - \text{diag}[Q_{*,*} - K_{*,*}] \end{bmatrix}\right). \quad (25)$$

7. The Partially Independent Training Conditional (PITC) Approximation

In the previous section we saw how to improve the DTC approximation by approximating the training conditional with an independent distribution, i.e. one with a diagonal covariance matrix. In this section we will further improve the approximation (while remaining computationally attractive) by extending the training conditional to have a block diagonal covariance:

$$q_{\text{PITC}}(\mathbf{f}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \text{blockdiag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}}]), \quad \text{and} \quad q_{\text{PITC}}(\mathbf{f}_*|\mathbf{u}) = p(\mathbf{f}_*|\mathbf{u}). \quad (26)$$

where $\text{blockdiag}[A]$ is a block diagonal matrix (where the blocking structure is not explicitly stated). We represent graphically the PITC approximation in Figure 3. Developing this analogously to the FITC approximation from the previous section, we get the joint prior

$$q_{\text{PITC}}(\mathbf{f}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \text{blockdiag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}\right), \quad (27)$$

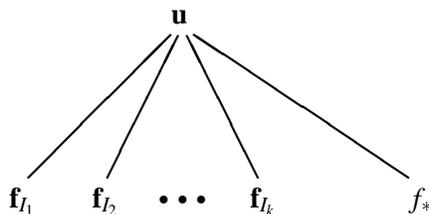


Figure 3: Graphical representation of the PITC approximation. The set of latent function values \mathbf{f}_{I_i} indexed by the the set of indices I_i is fully connected. The PITC differs from FITC (see graph in Fig. 2) in that conditional independence is now between the k groups of training latent function values. This corresponds to the block diagonal approximation to the true training conditional given in (26).

and the predictive distribution is identical to (24), except for the alternative definition of $\Lambda = \text{blockdiag}[K_{\mathbf{f},\mathbf{f}} - Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I]$. An identical expression was obtained by Schwaighofer and Tresp (2003, Sect. 3), developing from the original Bayesian committee machine (BCM) by Tresp (2000). The relationship to the FITC was pointed out by Lehel Csató. The BCM was originally proposed as a transductive learner (i.e. where the *test* inputs have to be known before training), and the inducing inputs $X_{\mathbf{u}}$ were chosen to be the test inputs. We discuss transduction in detail in the next section.

It is important to realize that the BCM proposes two orthogonal ideas: first, the block diagonal structure of the partially independent training conditional, and second setting the inducing inputs to be the test inputs. These two ideas can be used independently and in Section 8 we propose using the first without the second.

The computational complexity of the PITC approximation depends on the blocking structure imposed in (26). A reasonable choice, also recommended by Tresp (2000) may be to choose $k = n/m$ blocks, each of size $m \times m$. The computational complexity remains $O(nm^2)$. Since in the PITC model the covariance is computed differently for training and test cases

Remark 8 *The PITC approximation does not correspond exactly to a Gaussian process.*

This is because computing covariances requires knowing whether points are from the training- or test-set, (27). One can obtain a Gaussian process from the PITC by extending the partial conditional independence assumption to the test conditional, as we did in Remark 7 for the FITC.

8. Transduction and Augmentation

The idea of transduction is that one should restrict the goal of learning to prediction on a pre-specified set of test cases, rather than trying to learn an entire function (induction) and then evaluate it at the test inputs. There may be no universally agreed upon definition of transduction. In this paper we use

Definition 9 *Transduction occurs only if the predictive distribution depends on other test inputs.*

This operational definition excludes models for which there exist an equivalent inductive counterpart. According to this definition, it is irrelevant when the bulk of the computation takes place.

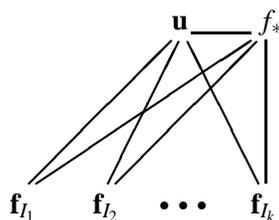


Figure 4: Two views on Augmentation. One view is to see that the test latent function value f_* is now part of the inducing variables \mathbf{u} and therefore has access to the training latent function values. An equivalent view is to consider that we have dropped the assumption of conditional independence between f_* and the training latent function values. Even if f_* has now direct access to each of the training f_i , these still need to go through \mathbf{u} to talk to each other if they fall in conditionally independent blocks. We have in this figure decided to recycle the graph for PITC from Figure 3 to show that all approximations we have presented can be augmented, irrespective of what the approximation for the training conditional is.

There are several different possible motivations for transduction: 1) transduction is somehow easier than induction (Vapnik, 1995), 2) the test inputs may reveal important information, which should be used during training. This motivation drives models in semi-supervised learning (studied mostly in the context of classification) and 3) for approximate algorithms one may be able to limit the discrepancies of the approximation at the test points.

For exact GP models it seems that the first reason doesn't really apply. If you make predictions at the test points that are consistent with a GP, then it is trivial inside the GP framework to extend these to any other input points, and in effect we have done induction.

The second reason seems more interesting. However, in a standard GP setting, it is a consequence of the consistency property, see Remark 2, that predictions at one test input are independent of the location of any other test inputs. Therefore transduction can not be married with exact GPs:

Remark 10 *Transduction can not occur in exact Gaussian process models.*

Whereas this holds for the usual setting of GPs, it could be different in non-standard situations where e.g. the covariance function depends on the empirical input densities.

Transduction can occur in the sparse approximation to GPs, by making the choice of inducing variables depend on the test inputs. The BCM from the previous section, where $X_{\mathbf{u}} = X_*$ (where X_* are the test inputs) is an example of this. Since the inducing variables are connected to all other nodes (see Figure 3) we would expect the approximation to be good at $\mathbf{u} = \mathbf{f}_*$, which is what we care about for predictions, relating to reason 3) above. While this reasoning is sound, it is not necessarily a sufficient consideration for getting a good model. The model has to be able to simultaneously explain the training targets as well and if the choice of \mathbf{u} makes this difficult, the posterior at the points of interest may be distorted. Thus, the choice of \mathbf{u} should be governed by the ability to model the conditional of the latents given the inputs, and not solely by the density of the (test) inputs.

The main drawback of transduction is that by its nature it doesn't provide a predictive model in the way inductive models do. In the usual GP model one can do the bulk of the computation

involved in the predictive distributions (e.g. matrix inversion) *before* seeing the test cases, enabling fast computation of test predictions.

It is interesting that whereas other methods spend much effort trying to optimize the inducing variables, the BCM simply uses the test set. The quality of the BCM approximation depends then on the particular location of the test inputs, upon which one usually does not have any control. We now see that there may be a better method, eliminating the drawback of transduction, namely use the PITC approximation, but choose the \mathbf{u} 's carefully (see Section 9), don't just use the test set.

8.1 Augmentation

An idea closely related to transduction, but not covered by our definition, is augmentation, which in contrast to transduction is done individually for each test case. Since in the previous sections, we haven't assumed anything about \mathbf{u} , we can simply augment the set of inducing variables by f_* (i.e. have one additional inducing variable equal to the current test latent), and see what happens in the predictive distributions for the different methods. Let's first investigate the consequences for the test conditional from (9b). Note, the interpretation of the covariance matrix $K_{*,*} - Q_{*,*}$ was "the prior covariance minus the information which \mathbf{u} provides about f_* ". It is clear that the augmented \mathbf{u} (with f_*) provides all possible information about f_* , and consequently $Q_{*,*} = K_{*,*}$. An equivalent view on augmentation is that the assumption of conditional independence between f_* and \mathbf{f} is dropped. This is seen trivially by adding edges between f_* and the f_i in the graphical model, Figure 4.

Augmentation was originally proposed by Rasmussen (2002), and applied in detail to the SoR with RBF covariance by Quiñonero-Candela (2004). Because the SoR is a finite linear model, and the basis functions are local (Gaussian bumps), the predictive distributions can be very misleading. For example, when making predictions far away from the center of any basis function, all basis functions have insignificant magnitudes, and the prediction (averaged over the posterior) will be close to zero, with very small error-bars; this is the opposite of the desired behaviour, where we would expect the error-bars to *grow* as we move away from the training cases. Here augmentation makes a particularly big difference turning the nonsensical predictive distribution into a reasonable one, by ensuring that there is always a basis function centered on the test case. Compare the non-augmented to the augmented SoR in Figure 5. An analogous Gaussian process based finite linear model that has recently been healed by augmentation is the relevance vector machine (Rasmussen and Quiñonero-Candela, 2005).

Although augmentation was initially proposed for a narrow set of circumstances, it is easily applied to any of the approximations discussed. Of course, augmentation doesn't make any sense for an exact, non-degenerate Gaussian process model (a GP with a covariance function that has a feature-space which is infinite dimensional, i.e. with basis functions *everywhere*).

Remark 11 *A full non-degenerate Gaussian process cannot be augmented,*

since the corresponding \mathbf{f}_* would already be connected to all other variables in the graphical model. But augmentation *does* make sense for sparse approximations to GPs.

The more general process view on augmentation has several advantages over the basis function view. It is not completely clear from the basis function view, which basis function should be used for augmentation. For example, Rasmussen and Quiñonero-Candela (2005) successfully apply augmentation using basis functions that have a zero contribution at the test location! In the process view

however, it seems clear that one would chose the additional inducing variable to be f_* , to minimize the effects of the approximations.

Let us compute the effective prior for the *augmented* SoR. Given that f_* is in the inducing set, the test conditional is not an approximation and we can rewrite the integral leading to the effective prior:

$$q_{\text{ASoR}}(\mathbf{f}_*, \mathbf{f}) = \int q_{\text{SoR}}(\mathbf{f}|f_*, \mathbf{u}) p(f_*, \mathbf{u}) \mathbf{d}\mathbf{u}. \quad (28)$$

It is interesting to notice that this is also the effective prior that would result from augmenting the DTC approximation, since $q_{\text{SoR}}(\mathbf{f}|f_*, \mathbf{u}) = q_{\text{DTC}}(\mathbf{f}|f_*, \mathbf{u})$.

Remark 12 *Augmented SoR (ASoR) is equivalent to augmented DTC (ADTC).*

Augmented DTC only differs from DTC in the additional presence of f_* among the inducing variables in the training conditional. We can only expect augmented DTC to be a more accurate approximation than DTC, since adding an additional inducing variable can only help capture information from \mathbf{y} . Therefore

Remark 13 *DTC is a less accurate (but cheaper) approximation than augmented SoR.*

We saw previously in Section 5 that the DTC approximation does not suffer from the nonsensical predictive variances of the SoR. The equivalence between the augmented SoR and augmented DTC is another way of seeing how augmentation reverses the misbehaviour of SoR. The predictive distribution of the augmented SoR is obtained by adding f_* to \mathbf{u} in (20).

Prediction with an augmented sparse model comes at a higher computational cost, since now f_* directly interacts with all of \mathbf{f} and not just with \mathbf{u} . For each new test case, updating the augmented Σ in the predictive equation (for example (20b) for DTC) implies computing the vector matrix product $K_{*,\mathbf{f}}K_{\mathbf{f},\mathbf{u}}$ with complexity $O(nm)$. This is clearly higher than the $O(m)$ for the mean, and $O(m^2)$ for the predictive distribution of all the non-augmented methods we have discussed.

Augmentation seems to be only really necessary for methods that make a severe approximation to the test conditional, like the SoR. For methods that make little or no approximation to the test conditional, it is difficult to predict the degree to which augmentation would help. However, one can see by giving f_* access to all of the training latent function values in \mathbf{f} , one would expect augmentation to give less under-confident predictive distributions near the training data. Figure 5 clearly shows that augmented DTC (equivalent to augmented SoR) has a superior predictive distribution (both mean and variance) than standard DTC. Note however that in the figure we have purposely chosen a too short lengthscale to enhance visualization. Quantitatively, this superiority was experimentally assessed by Quiñonero-Candela (2004, Table 3.1). Augmentation hasn't been compared to the more advanced approximations FITC and PITC, and the figure would change in the more realistic scenario where the inducing inputs and hyperparameters are learnt (Snelson and Ghahramani, 2006).

Transductive methods like the BCM can be seen as joint augmentation, and one could potentially use it for any of the methods presented. It seems that the good performance of the BCM could essentially stem from augmentation, the presence of the *other* test inputs in the inducing set being probably of little benefit. Joint augmentation might bring some computational advantage, but won't change the scaling: note that augmenting m times at a cost of $O(nm)$ apiece implies the same $O(nm^2)$ total cost as the jointly augmented BCM.

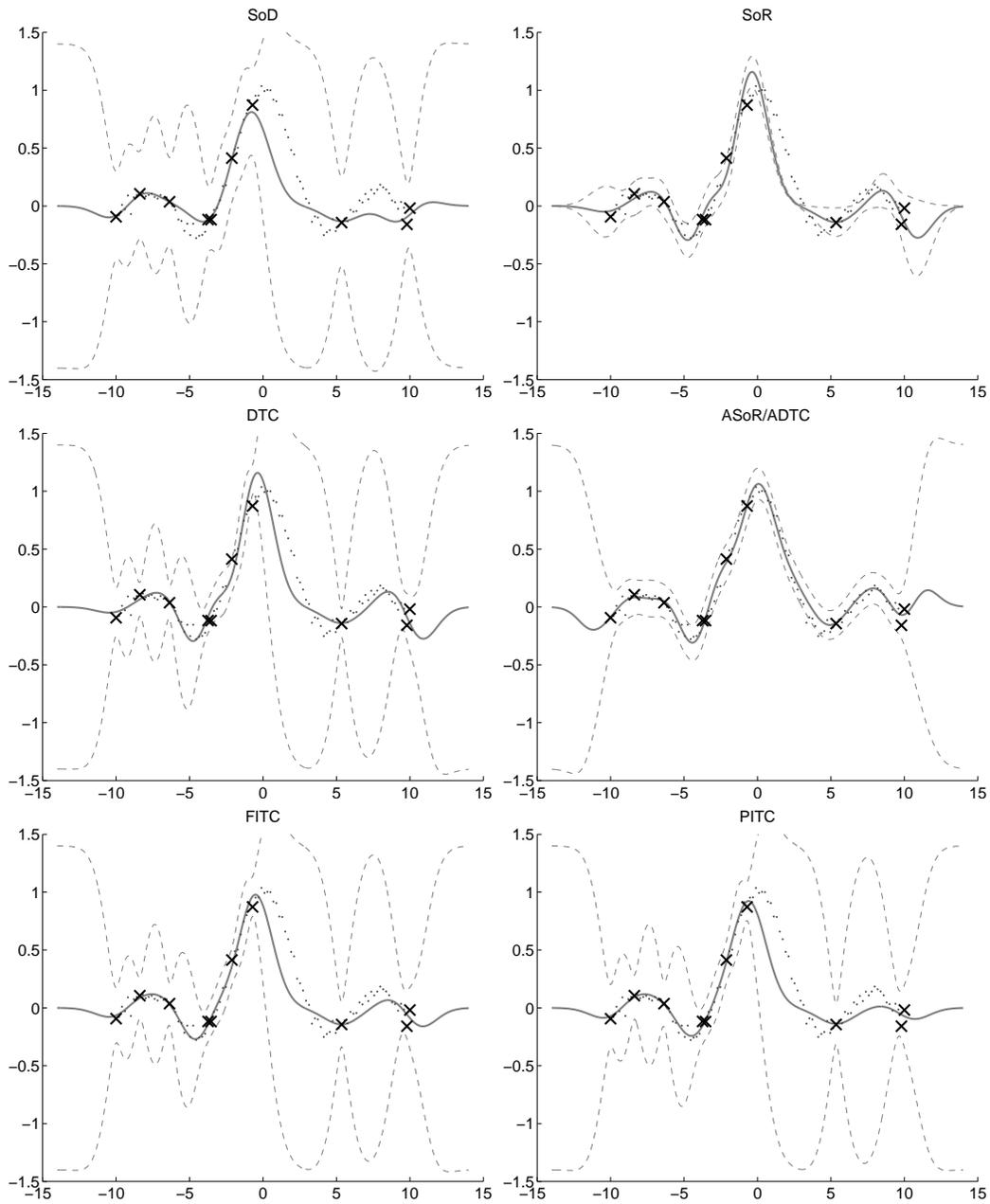


Figure 5: Toy example with identical covariance function and hyperparameters. The squared exponential covariance function is used, and a slightly too short lengthscale is chosen on purpose to emphasize the different behaviour of the predictive uncertainties. The dots are the training points, the crosses are the targets corresponding to the inducing inputs, randomly selected from the training set. The solid line is the mean of the predictive distribution, and the dotted lines show the 95% confidence interval of the predictions. Augmented DTC (ADTC) is equivalent to augmented SoR (ASoR), see Remark 12.

9. On the Choice of the Inducing Variables

We have until now assumed that the inducing inputs $X_{\mathbf{u}}$ were given. Traditionally, sparse models have very often been built upon a carefully chosen subset of the training inputs. This concept is probably best exemplified in the popular support vector machine (Cortes and Vapnik, 1995). In sparse Gaussian processes it has also been suggested to select the inducing inputs $X_{\mathbf{u}}$ from among the training inputs. Since this involves a prohibitive combinatorial optimization, greedy optimization approaches have been suggested using various selection criteria like online learning (Csató and Opper, 2002), greedy posterior maximization (Smola and Bartlett, 2001), maximum information gain (Seeger et al., 2003), matching pursuit (Keerthi and Chu, 2006), and probably more. As discussed in the previous section, selecting the inducing inputs from among the test inputs has also been considered in transductive settings. Recently, Snelson and Ghahramani (2006) have proposed to relax the constraint that the inducing variables must be a subset of training/test cases, turning the discrete selection problem into one of continuous optimization. One may hope that finding a good solution is easier in the continuous than the discrete case, although finding the global optimum is intractable in both cases. And perhaps the less restrictive choice can lead to better performance in very sparse models.

Which optimality criterion should be used to set the inducing inputs? Departing from a fully Bayesian treatment which would involve defining priors on $X_{\mathbf{u}}$, one could maximize the marginal likelihood (also called the evidence) with respect to $X_{\mathbf{u}}$, an approach also followed by Snelson and Ghahramani (2006). Each of the approximate methods proposed involves a different effective prior, and hence its own particular effective marginal likelihood conditioned on the inducing inputs

$$q(\mathbf{y}|X_{\mathbf{u}}) = \iint p(\mathbf{y}|\mathbf{f}) q(\mathbf{f}|\mathbf{u}) p(\mathbf{u}|X_{\mathbf{u}}) d\mathbf{u} d\mathbf{f} = \int p(\mathbf{y}|\mathbf{f}) q(\mathbf{f}|X_{\mathbf{u}}) d\mathbf{f}, \quad (29)$$

which of course is independent of the test conditional. We have in the above equation explicitly conditioned on the inducing inputs $X_{\mathbf{u}}$. Using Gaussian identities, the effective marginal likelihood is very easily obtained by adding a ridge $\sigma_{\text{noise}}^2 I$ (from the likelihood) to the covariance of effective prior on \mathbf{f} . Using the appropriate definitions of Λ , the log marginal likelihood becomes

$$\log q(\mathbf{y}|X_{\mathbf{u}}) = -\frac{1}{2} \log |\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda| - \frac{1}{2} \mathbf{y}^{\top} (\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda)^{-1} \mathbf{y} - \frac{n}{2} \log(2\pi), \quad (30)$$

where $\Lambda_{\text{SOR}} = \Lambda_{\text{DTC}} = \sigma_{\text{noise}}^2 I$, $\Lambda_{\text{FITC}} = \text{diag}[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{Q}_{\mathbf{f},\mathbf{f}}] + \sigma_{\text{noise}}^2 I$, and $\Lambda_{\text{PTC}} = \text{blockdiag}[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{Q}_{\mathbf{f},\mathbf{f}}] + \sigma_{\text{noise}}^2 I$. The computational cost of the marginal likelihood is $O(nm^2)$ for all methods, that of its gradient with respect to one element of $X_{\mathbf{u}}$ is $O(nm)$. This of course implies that the complexity of computing the gradient wrt. to the whole of $X_{\mathbf{u}}$ is $O(dnm^2)$, where d is the dimension of the input space.

It has been proposed to maximize the effective posterior instead of the effective marginal likelihood (Smola and Bartlett, 2001). However this is potentially dangerous and can lead to overfitting. Maximizing the whole evidence instead is sound and comes at an identical computational cost (for a deeper analysis see Quiñonero-Candela, 2004, Sect. 3.3.5 and Fig. 3.2).

The marginal likelihood has traditionally been used to learn the hyperparameters of GPs in the non fully Bayesian treatment (see for example Williams and Rasmussen, 1996). For the sparse approximations presented here, once you are learning $X_{\mathbf{u}}$ it is straightforward to allow for learning hyperparameters (of the covariance function) during the same optimization, and there is no need to interleave optimization of \mathbf{u} with learning of the hyperparameters as it has been proposed for example by Seeger et al. (2003).

10. Other Methods

In this section we briefly mention two approximations which don't fit in our unifying scheme, since one doesn't correspond to a proper probabilistic model, and the other one uses a particular construction for the covariance function, rather than allowing any general covariance function.

10.1 The Nyström Approximation

The Nyström Approximation for speeding up GP regression was originally proposed by Williams and Seeger (2001), and then questioned by Williams et al. (2002). Like SoR and DTC, the Nyström Approximation for GP regression approximates the prior covariance of \mathbf{f} by $Q_{\mathbf{f},\mathbf{f}}$. However, unlike these methods, the Nyström Approximation is *not* based on a generative probabilistic model. The prior covariance between f_* and \mathbf{f} is taken to be exact, which is *inconsistent* with the prior covariance on \mathbf{f} :

$$q(\mathbf{f}, \mathbf{f}_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & K_{\mathbf{f},*} \\ K_{*\mathbf{f}} & K_{*,*} \end{bmatrix}\right). \quad (31)$$

As a result we cannot derive this method from our unifying framework, nor represent it with a graphical model. Worse, the resulting prior covariance matrix is not even guaranteed to be positive definite, allowing the predictive variances to be negative. Notice that replacing $K_{\mathbf{f},*}$ by $Q_{\mathbf{f},*}$ in (31) is enough to make the prior covariance positive definite, and one obtains the DTC approximation.

Remark 14 *The Nyström Approximation does not correspond to a well-formed probabilistic model.*

Ignoring any quibbles about positive definiteness, the predictive distribution of the Nyström Approximation is given by:

$$p(f_* | \mathbf{y}) = \mathcal{N}(K_{\mathbf{f},*}^\top [Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I]^{-1} \mathbf{y}, K_{*,*} - K_{\mathbf{f},*}^\top [Q_{\mathbf{f},\mathbf{f}} + \sigma_{\text{noise}}^2 I]^{-1} K_{\mathbf{f},*}), \quad (32)$$

but the predictive variance is not guaranteed to be positive. The computational cost is $O(nm^2)$.

10.2 The Relevance Vector Machine

The relevance vector machine, introduced by Tipping (2001), is a finite linear model with an independent Gaussian prior imposed on the weights. For any input \mathbf{x}_* , the corresponding function output is given by:

$$f_* = \phi_* \mathbf{w}, \quad \text{with} \quad p(\mathbf{w} | A) = \mathcal{N}(0, A), \quad (33)$$

where $\phi_* = [\phi_1(\mathbf{x}), \dots, \phi_m(\mathbf{x})]$ is the (row) vector of responses of the m basis functions, and $A = \text{diag}(\alpha_1, \dots, \alpha_m)$ is the diagonal matrix of joint prior precisions (inverse variances) of the weights. The α_i are learnt by maximizing the RVM evidence (obtained by also assuming Gaussian additive iid. noise, see (1)), and for the typical case of rich enough sets of basis functions many of the precisions go to infinity effectively pruning out the corresponding weights (for a very interesting analysis see Wipf et al., 2004). The RVM is thus a sparse method and the surviving basis functions are called *relevance vectors*.

Note that since the RVM is a finite linear model with Gaussian priors on the weights, it can be seen as a Gaussian process:

Remark 15 *The RVM is equivalent to a degenerate Gaussian process with covariance function*
 $k_{\text{RVM}}(\mathbf{x}_i, \mathbf{x}_j) = \phi_i A^{-1} \phi_j^\top = \sum_{k=1}^m \alpha_k^{-1} \phi_k(\mathbf{x}_i) \phi_k(\mathbf{x}_j),$

Method	$q(\mathbf{f}_* \mathbf{u})$	$q(\mathbf{f} \mathbf{u})$	joint prior covariance	GP?
GP	exact	exact	$\begin{bmatrix} K_{\mathbf{f},\mathbf{f}} & K_{\mathbf{f},*} \\ K_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	✓
SoR	determ.	determ.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & Q_{*,*} \end{bmatrix}$	✓
DTC	exact	determ.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	
FITC	(exact)	fully indep.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \text{diag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	(✓)
PITC	exact	partially indep.	$\begin{bmatrix} Q_{\mathbf{f},\mathbf{f}} - \text{blokdiag}[Q_{\mathbf{f},\mathbf{f}} - K_{\mathbf{f},\mathbf{f}}] & Q_{\mathbf{f},*} \\ Q_{*,\mathbf{f}} & K_{*,*} \end{bmatrix}$	

Table 1: Summary of the way approximations are built. All these methods are detailed in the previous sections. The initial cost and that of the mean and variance per test case are respectively n^2 , n and n^2 for the exact GP, and nm^2 , m and m^2 for all other methods. The “GP?” column indicates whether the approximation is equivalent to a GP. For FITC see Remark 7.

as was also pointed out by Tipping (2001, eq. (59)). Whereas all sparse approximations we have presented until now are totally independent of the choice of covariance function, for the RVM this choice is restricted to covariance functions that can be expressed as finite expansions in terms of some basis functions. Being degenerate GPs in exactly the same way as the SoR (presented in Section 4), the RVM does also suffer from unreasonable predictive variances. Rasmussen and Quiñonero-Candela (2005) show that the predictive distributions of RVMs can also be healed by augmentation, see Section 8. Once the α_i have been learnt, denoting by m the number of surviving relevance vectors, the complexity of computing the predictive distribution of the RVM is $O(m)$ for mean and $O(m^2)$ for the variance.

RVMs are often used with radial basis functions centered on the training inputs. One potentially interesting extension to the RVM would be to *learn* the locations of the centers of the basis functions, in the same way as proposed by Snelson and Ghahramani (2006) for the FITC approximation, see Section 6. This is a curious reminiscence of learning the centers in RBF Networks.

11. Conclusions

We have provided a unifying framework for sparse approximations to Gaussian processes for regression. Our approach consists of two steps, first 1) we recast the approximation in terms of approximations to the prior, and second 2) we introduce inducing variables \mathbf{u} and the idea of conditional independence given \mathbf{u} . We recover all existing sparse methods by making further simplifications of the covariances of the training and test conditionals, see Table 1 for a summary.

Previous methods were presented based on different approximation paradigms (e.g. likelihood approximations, projection methods, matrix approximations, minimization of Kullback-Leibler divergence, etc), making direct comparison difficult. Under our unifying view we deconstruct methods, making it clear which building blocks they are based upon. For example, the SGPP by Snelson

and Ghahramani (2006) contains two ideas, 1) a likelihood approximation and 2) the idea of varying the inducing inputs continuously; these two ideas could easily be used independently, and incorporated in other methods. Similarly, the BCM by Tresp (2000) contains two independent ideas 1) a block diagonal assumption, and 2) the (transductive) idea of choosing the test inputs as the inducing variables. Finally we note that although all three ideas of 1) transductively setting $\mathbf{u} = \mathbf{f}_*$, 2) augmentation and 3) continuous optimization of $X_{\mathbf{u}}$ have been proposed in very specific settings, in fact they are completely general ideas, which can be applied to any of the approximation schemes considered.

We have ranked the approximation according to how close they are to the corresponding full GP. However, the performance in practical situations may not always follow this theoretical ranking since the approximations might exhibit properties (not present in the full GP) which may be particularly suitable for specific datasets. This may make the interpretation of empirical comparisons challenging. A further complication arises when adding the necessary heuristics for turning the theoretical constructs into practical algorithms. We have not described full algorithms in this paper, but are currently working on a detailed empirical study (in preparation, see also Rasmussen and Williams, 2006, chapter 8).

We note that the order of the computational complexity is identical for all the methods considered, $O(nm^2)$. This highlights that there is no computational excuse for using gross approximations, such as assuming deterministic relationships, in particular one should probably think twice before using SoR or even DTC. Although augmentation has attractive predictive properties, it is computationally expensive. It remains unclear whether augmentation could be beneficial on a fixed computational budget.

We have only considered the simpler case of regression in this paper, but sparseness is also commonly sought in classification settings. It should not be difficult to cast probabilistic approximation methods such as Expectation Propagation (EP) or the Laplace method (for a comparison, see Kuss and Rasmussen, 2005) into our unifying framework.

Our analysis suggests that a new interesting approximation would come from combining the best possible approximation (PITC) with the most powerful selection method for the inducing inputs. This would correspond to a non-transductive version of the BCM. We would evade the necessity of knowing the test set before doing the bulk of the computation, and we could hope to supersede the superior performance reported by Snelson and Ghahramani (2006) for very sparse approximations.

Acknowledgments

Thanks to Neil Lawrence for arranging the 2005 Gaussian Process Round Table meeting in Sheffield, which provided much inspiration to this paper. Special thanks to Olivier Chapelle, Lehel Csató, Zoubin Ghahramani, Matthias Seeger, Ed Snelson and Chris Williams for helpful discussions, and to three anonymous reviewers. Both authors were supported by the German Research Council (DFG) through grant RA 1030/1. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

Appendix A. Gaussian and Matrix Identities

In this appendix we provide identities used to manipulate matrices and Gaussian distributions throughout the paper. Let \mathbf{x} and \mathbf{y} be jointly Gaussian

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix}\right), \quad (34)$$

then the marginal and the conditional are given by

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, A), \quad \text{and} \quad \mathbf{x}|\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_x + CB^{-1}(\mathbf{y} - \boldsymbol{\mu}_y), A - CB^{-1}C^\top) \quad (35)$$

Also, the product of a Gaussian in \mathbf{x} with a Gaussian in a linear projection $P\mathbf{x}$ is again a Gaussian, although unnormalized

$$\mathcal{N}(\mathbf{x}|\mathbf{a}, A) \mathcal{N}(P\mathbf{x}|\mathbf{b}, B) = z_c \mathcal{N}(\mathbf{x}|\mathbf{c}, C), \quad (36)$$

where

$$C = (A^{-1} + P^\top B^{-1}P)^{-1}, \quad c = C(A^{-1}\mathbf{a} + P^\top B^{-1}\mathbf{b}).$$

The normalizing constant z_c is gaussian in the means \mathbf{a} and \mathbf{b} of the two Gaussians:

$$z_c = (2\pi)^{-\frac{m}{2}} |B + PAP^\top|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{b} - P\mathbf{a})^\top (B + PAP^\top)^{-1}(\mathbf{b} - P\mathbf{a})\right). \quad (37)$$

The matrix inversion lemma, also known as the Woodbury, Sherman & Morrison formula states that:

$$(Z + UWV^\top)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^\top Z^{-1}U)^{-1}V^\top Z^{-1}, \quad (38)$$

assuming the relevant inverses all exist. Here Z is $n \times n$, W is $m \times m$ and U and V are both of size $n \times m$; consequently if Z^{-1} is known, and a low rank (ie. $m < n$) perturbation are made to Z as in left hand side of eq. (38), considerable speedup can be achieved.

References

- Corinna Cortes and Vladimir Vapnik. Support-vector network. *Machine Learning*, 20(3):273–297, 1995.
- Lehel Csató and Manfred Opper. Sparse online Gaussian processes. *Neural Computation*, 14(3): 641–669, 2002.
- Sathya Keerthi and Wei Chu. A Matching Pursuit approach to sparse Gaussian process regression. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, Massachusetts, 2006. The MIT Press.
- Malte Kuss and Carl Edward Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, pages 1679–1704, 2005.
- Joaquin Quiñero-Candela. *Learning with Uncertainty – Gaussian Processes and Relevance Vector Machines*. PhD thesis, Technical University of Denmark, Lyngby, Denmark, 2004.
- Carl Edward Rasmussen. Reduced rank Gaussian process learning. Technical report, Gatsby Computational Neuroscience Unit, UCL, 2002.

- Carl Edward Rasmussen and Joaquin Quiñonero-Candela. Healing the relevance vector machine by augmentation. In *International Conference on Machine Learning*, 2005.
- Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT press, 2006.
- Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate Gaussian process regression. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 953–960, Cambridge, Massachusetts, 2003. The MIT Press.
- Matthias Seeger, Christopher K. I. Williams, and Neil Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In Christopher M. Bishop and Brendan J. Frey, editors, *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2003.
- Bernhard W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *J. Roy. Stat. Soc. B*, 47(1):1–52, 1985. (with discussion).
- Alexander J. Smola and Peter L. Bartlett. Sparse greedy Gaussian process regression. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625, Cambridge, Massachusetts, 2001. The MIT Press.
- Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, Massachusetts, 2006. The MIT Press.
- Michael E. Tipping. Sparse Bayesian learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001.
- Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.
- Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.
- Grace Wahba, Xiwu Lin, Fangyu Gao, Dong Xiang, Ronald Klein, and Barbara Klein. The bias-variance tradeoff and the randomized GACV. In Michael S. Kerns, Sara A. Solla, and David A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 620–626, Cambridge, Massachusetts, 1999. The MIT Press.
- Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520, Cambridge, Massachusetts, 1996. The MIT Press.
- Christopher K. I. Williams, Carl Edward Rasmussen, Anton Schwaighofer, and Volker Tresp. Observations of the Nyström method for Gaussian process prediction. Technical report, University of Edinburgh, Edinburgh, Scotland, 2002.

Christopher K. I. Williams and Mathias Seeger. Using the Nyström method to speed up kernel machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688, Cambridge, Massachusetts, 2001. The MIT Press.

David Wipf, Jason Palmer, and Bhaskar Rao. Perspectives on sparse Bayesian learning. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, Massachusetts, 2004. The MIT Press.

What's Strange About Recent Events (WSARE): An Algorithm for the Early Detection of Disease Outbreaks

Weng-Keen Wong

*School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97330, USA*

WONG@EECS.OREGONSTATE.EDU

Andrew Moore

*School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA*

AWM@CS.CMU.EDU

Gregory Cooper

*Center For Biomedical Informatics
University of Pittsburgh
Pittsburgh, PA 15213, USA*

GFC@CBML.PITT.EDU

Michael Wagner

MMW@CBML.PITT.EDU

Editor: Dale Schuurmans

Abstract

Traditional biosurveillance algorithms detect disease outbreaks by looking for peaks in a univariate time series of health-care data. Current health-care surveillance data, however, are no longer simply univariate data streams. Instead, a wealth of spatial, temporal, demographic and symptomatic information is available. We present an early disease outbreak detection algorithm called What's Strange About Recent Events (WSARE), which uses a multivariate approach to improve its timeliness of detection. WSARE employs a rule-based technique that compares recent health-care data against data from a baseline distribution and finds subgroups of the recent data whose proportions have changed the most from the baseline data. In addition, health-care data also pose difficulties for surveillance algorithms because of inherent temporal trends such as seasonal effects and day of week variations. WSARE approaches this problem using a Bayesian network to produce a baseline distribution that accounts for these temporal trends. The algorithm itself incorporates a wide range of ideas, including association rules, Bayesian networks, hypothesis testing and permutation tests to produce a detection algorithm that is careful to evaluate the significance of the alarms that it raises.

Keywords: anomaly detection, syndromic surveillance, biosurveillance, Bayesian networks, applications

1. Introduction

Detection systems inspect routinely collected data for anomalies and raise an alert upon discovery of any significant deviations from the norm. For example, Fawcett and Provost (1997) detect cellular phone fraud by monitoring changes to a cell phone user's typical calling behavior. In intrusion detection systems, anomalies in system events might indicate a possible breach of security (Warren-

der et al., 1999). In a similar manner, we would like to tackle the problem of early disease outbreak detection, in which the disease outbreak can be due to either natural causes or a bioterrorist attack.

One of the challenges for early disease outbreak detection is finding readily available data that contains a useful signal (Tsui et al., 2001). Data sources that require definitive diagnosis of the disease, such as lab reports, can often be obtained several days to weeks after the samples are submitted. By that point, the outbreak may have already escalated into a large scale epidemic. Instead of waiting for definite diagnostic data, we can monitor pre-diagnosis data, such as the symptoms exhibited by patients at an Emergency Department (ED). In doing so, we risk increasing the false positive rate, such as mistakenly attributing an increase in patients exhibiting respiratory problems to an anthrax attack rather than to influenza. Nevertheless, we have a potential gain in timeliness of detection. This type of surveillance of pre-diagnosis data is commonly referred to as *syndromic surveillance* (Mostashari and Hartman, 2003; Sosin, 2003).

In our syndromic surveillance infrastructure, we have real-time access to a database of emergency department (ED) cases from several hospitals in a city. Each record in this multivariate database contains information about the individual who is admitted to the ED. This information includes fields such as age, gender, symptoms exhibited, home zip code, work zip code, and time of arrival at the ED. In accordance with the HIPAA Privacy Rule (45 CFR Parts 160 through 164, 2003), personal identifying information, such as patient names, addresses, and identification numbers are removed from the data set used in this research. When a severe epidemic sweeps through a region, there will obviously be extreme perturbations in the number of ED visits. While these dramatic upswings are easily noticed during the late stages of an epidemic, the challenge is to detect the outbreak during its early stages and mitigate its effects. We would also like to detect outbreaks that are more subtle than a large scale epidemic as early as possible.

Although we have posed our problem in an anomaly detection framework, traditional anomaly detection algorithms are inappropriate for this domain. In the traditional approach, a probabilistic model of the baseline data is built using techniques such as neural nets (Bishop, 1994) or a mixture of naive Bayes submodels (Hamerly and Elkan, 2001). Anomalies are identified as individual data points with a rare attribute or rare combination of attributes. If we apply traditional anomaly detection to our ED data, we would find, for example, a patient that is over a hundred years old living in a sparsely populated region of the city. These isolated outliers in attribute space are not at all indicative of a disease outbreak. Instead of finding such unusual isolated cases, we are interested in finding *anomalous patterns*, which are specific groups whose profile is anomalous relative to their typical profile. Thus, in our example of using ED records, if there is a dramatic upswing in the number of children from a particular neighborhood appearing in the ED with diarrhea, then an early detection system should raise an alarm.

Another common approach to early outbreak detection is to convert the multivariate ED database into a univariate time series by aggregating daily counts of a certain attribute or combination of attributes. For instance, a simple detector would monitor the daily number of people appearing in the ED. Many different algorithms can then be used to monitor this univariate surveillance data, including methods from Statistical Quality Control (Montgomery, 2001), time series models (Box and Jenkins, 1976), and regression techniques (Serfling, 1963). This technique works well if we know beforehand which disease to monitor, since we can improve the timeliness of detection by monitoring specific attributes of the disease. For example, if we are vigilant against an anthrax attack, we can concentrate our efforts on ED cases involving respiratory problems. In our situation, we need to perform non-specific disease monitoring because we do not know what disease to expect,

particularly in the case of a bioterrorist attack. Instead of monitoring health-care data for pre-defined patterns, we detect any significant anomalous patterns in the multivariate ED data. Furthermore, by taking a multivariate approach that inspects all available attributes in the data, particularly the temporal, spatial, demographic, and symptomatic attributes, we will show that such an approach can improve on the detection time of a univariate detection algorithm if the outbreak initially manifests itself as a localized cluster in attribute space.

Our approach to early disease outbreak detection uses a rule-based anomaly pattern detector called What's Strange About Recent Events (WSARE) (Wong et al., 2002, 2003). WSARE operates on discrete, multidimensional data sets with a temporal component. This algorithm compares recent data against a baseline distribution with the aim of finding rules that summarize significant patterns of anomalies. Each rule is made up of components of the form $X_i = V_i^j$, where X_i is the i th attribute and V_i^j is the j th value of that attribute. Multiple components are joined together by a logical AND. For example, a two component rule would be *Gender = Male AND Home Location = NW*. These rules should not be interpreted as rules from a logic-based system in which the rules have an antecedent and a consequent. Rather, these rules can be thought of as SQL SELECT queries because they identify a subset of the data having records with attributes that match the components of the rule. WSARE finds these subsets whose proportions have changed the most between recent data and the baseline.

We will present versions 2.0 and 3.0 of the WSARE algorithm. We will also briefly describe WSARE 2.5 in order to illustrate the strengths of WSARE 3.0. These three algorithms only differ in how they create the baseline distribution; all other steps in the WSARE framework remain identical. WSARE 2.0 and 2.5 use raw historical data from selected days as the baseline while WSARE 3.0 models the baseline distribution using a Bayesian network.

2. What's Strange About Recent Events

<i>November 2003</i>						
<i>Su</i>	<i>Mo</i>	<i>Tu</i>	<i>We</i>	<i>Th</i>	<i>Fr</i>	<i>Sa</i>
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						
<i>December 2003</i>						
<i>Su</i>	<i>Mo</i>	<i>Tu</i>	<i>We</i>	<i>Th</i>	<i>Fr</i>	<i>Sa</i>
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Figure 1: The baseline for WSARE 2.0 if the current day is December 30, 2003

The basic question asked by all detection systems is whether anything strange has occurred in recent events. This question requires defining what it means to be recent and what it means to be strange. Our algorithm considers all patient records falling on the current day under evaluation to be recent events. Note that this definition of recent is not restrictive – our approach is fully general and recent can be defined to include all events within some other time period such as over the last six hours. In order to define an anomaly, we need to establish the concept of something being normal. In WSARE version 2.0, baseline behavior is assumed to be captured by raw historical data from the same day of the week in order to avoid environmental effects such as weekend versus weekday differences in the number of ED cases. This baseline period must be chosen from a time period similar to the current day. This can be achieved by being close enough to the current day to capture any seasonal or recent trends. On the other hand, the baseline period must also be sufficiently distant from the current day. This distance is required in case an outbreak happens on the current day but it remains undetected. If the baseline period is too close to the current day, the baseline period will quickly incorporate the outbreak cases as time progresses. In the description of WSARE 2.0 below, we assume that baseline behavior is captured by records that are in the set *baseline_days*. Typically, *baseline_days* contains the days that are 35, 42, 49, and 56 days prior to the day under consideration. We would like to emphasize that this baseline period is only used as an example; it can be easily modified to another time period without major changes to our algorithm. In Section 3 we will illustrate how version 3.0 of WSARE automatically generates the baseline using a Bayesian network.

We will refer to the events that fit a certain rule for the current day as C_{recent} . Similarly, the number of cases matching the same rule from the baseline period will be called $C_{baseline}$. As an example, suppose the current day is Tuesday December 30, 2003. The baseline used for WSARE 2.0 will then be November 4, 11, 18 and 25 of 2003 as seen in Figure 1. These dates are all from Tuesdays in order to avoid day of week variations.

2.1 Overview of WSARE

Parameter Name	Description	Default value
<i>max_rule_components</i>	Maximum number of components to a rule	2
<i>num_randomizations</i>	Number of iterations to the randomization test	1000
α_{FDR}	The significance level of the False Discovery Rate	0.05
<i>baseline_days</i> (WSARE 2.0 only)	Days to be used for the baseline	35, 42, 49, and 56 days prior to current date
<i>environmental_attributes</i> (WSARE 2.5 and 3.0)	Attributes that account for temporal trends	Not applicable
<i>num_baseline_samples</i> (WSARE 3.0 only)	The number of sampled records from the baseline Bayesian network	10000

Table 1: The main parameters in WSARE

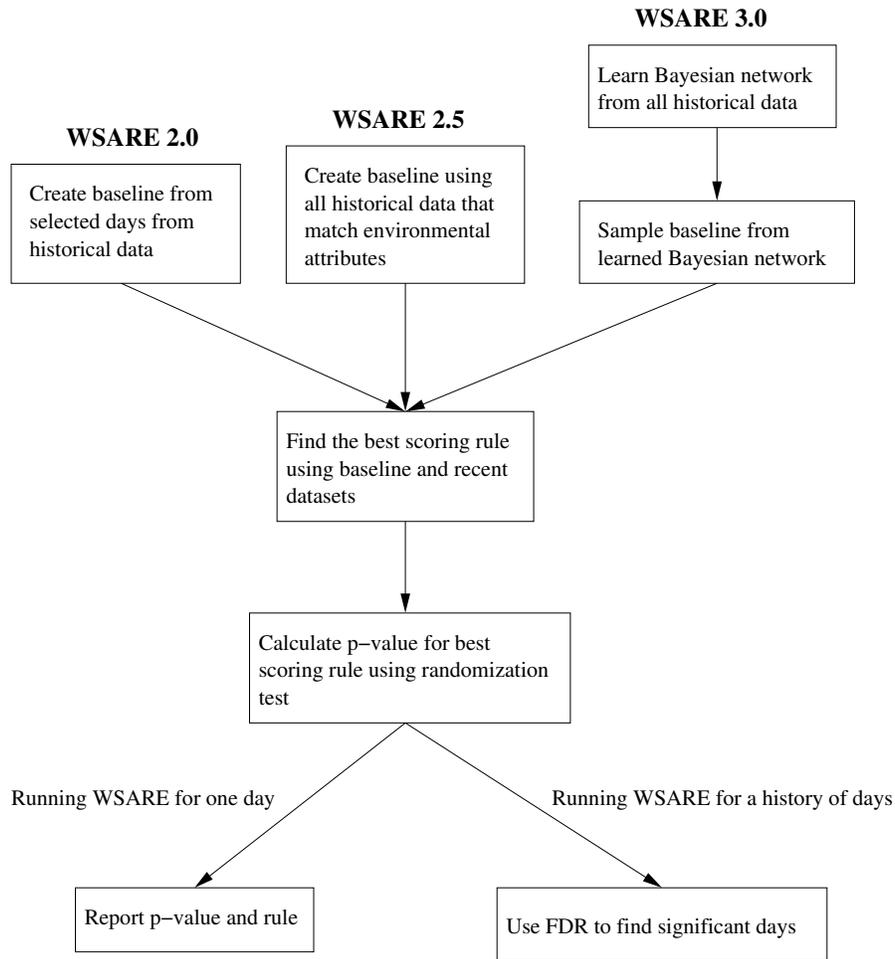


Figure 2: A schematic overview of the steps involved in the WSARE algorithms

We will begin this section with an overview of the general WSARE algorithm followed by a more detailed example. Figure 2 gives a pictorial overview of the three WSARE algorithms discussed in this paper. Note that the three algorithms differ only in how they create the baseline while all of the other steps remain identical. Table 1 describes the main parameters used by the WSARE algorithms.

WSARE first finds the best scoring rule over events occurring on the current day using a greedy search. The limit to the number of components in a rule is set to the parameter *max_rule_components*, which is typically set to be 2 for computational reasons although in Section 2.5 we describe a greedy procedure for *n* component rules. The score of a rule is determined by comparing the events on the current day against events in the past. More specifically, we are comparing if the ratio between certain events on the current day and the total number of events on the current day differ dramatically between the recent period and the past. Following the score calculation, the best rule for that day has its p-value estimated by a randomization test. The p-value for a rule is the likelihood of

finding a rule with as good a score under the hypothesis that the date and the other attributes are independent. The randomization-based p-value takes into account the effect of the multiple testing that occurs during the rule search. The number of iterations of the randomization test is determined by the parameter *num_randomizations*. If we are running the algorithm on a day-by-day basis we would end at this step. However, if we are looking at a history of days and we want to control for some level of false discoveries over this group of days, we would need the additional step of using the False Discovery Rate (FDR) method (Benjamini and Hochberg, 1995) to determine which of the p-values are significant. The days with significant p-values are returned as the anomalies.

2.2 One Component Rules

In order to illustrate this algorithm, suppose we have a large database of 1,000,000 ED records over a two-year span. This database contains roughly 1370 records a day. Suppose we treat all records within the last 24 hours as “recent” events. In addition, we can build a baseline data set out of all cases from exactly 35, 42, 49, and 56 days prior to the current day. We then combine the recent and baseline data to form a record subset called DB_i , which will have approximately 5000 records. The algorithm proceeds as follows. For each day i in the surveillance period, retrieve the records belonging to DB_i . We first consider all possible one-component rules. For every possible attribute-value combination, obtain the counts C_{recent} and $C_{baseline}$ from the data set DB_i . As an example, suppose the attribute under consideration is *Age Decile* for the ED case. There are 9 possible values for *Age Decile*, ranging from 0 to 8. We start with the rule $Age\ Decile = 3$ and count the number of cases for the current day i that have $Age\ Decile = 3$ and those that have $Age\ Decile \neq 3$. The cases from five to eight weeks ago are subsequently examined to obtain the counts for the cases matching the rule and those not matching the rule. The four values form a two-by-two contingency table such as the one shown in Table 2.

2.3 Scoring Each One Component Rule

The next step is to evaluate the “score” of the rule using a hypothesis test in which the null hypothesis is the independence of the row and column attributes of the two-by-two contingency table. In effect, the hypothesis test measures how different the distribution for C_{recent} is compared to that of $C_{baseline}$. This test will generate a p-value that determines the significance of the anomalies found by the rule. We will refer to this p-value as the *score* in order to distinguish this p-value from the p-value that is obtained later on from the randomization test. We use the Chi Square test for independence of variables whenever the counts in the contingency table do not violate the validity of the Chi Square test. However, since we are searching for anomalies, the counts in the contingency table frequently involve small numbers. In this case, we use Fisher’s exact test (Good, 2000) to find the score for each rule since the Chi Square test is an approximation to Fisher’s exact test when counts are large. Running Fisher’s exact test on Table 2 yields a score of 0.025939, which indicates that the count C_{recent} for cases matching the rule $Home\ Location = NW$ are very different from the count $C_{baseline}$. In biosurveillance, we are usually only interested in an increase in the number of certain records. As a result, we commonly use a one-sided Fisher’s exact test.

	C_{recent}	$C_{baseline}$
$Home\ Location = NW$	6	496
$Home\ Location \neq NW$	40	9504

Table 2: A Sample 2x2 Contingency Table

2.4 Two Component Rules

At this point, the best one component rule for a particular day has been found. We will refer to the best one component rule for day i as BR_i^1 . The algorithm then attempts to find the best two component rule for the day by adding on one extra component to BR_i^1 through a greedy search. This extra component is determined by supplementing BR_i^1 with all possible attribute-value pairs, except for the one already present in BR_i^1 , and selecting the resulting two component rule with the best score. Scoring is performed in the exact same manner as before, except the counts C_{recent} and $C_{baseline}$ are calculated by counting the records that match the two component rule. The best two-component rule for day i is subsequently found and we will refer to it as BR_i^2 .

Suppose BR_i^1 has as its first component the attribute-value pair $C_1 = V_1$. Furthermore, let BR_i^2 's components be $C_1 = V_1$ and $C_2 = V_2$. Adding the component $C_2 = V_2$ to BR_i^1 may not result in a better scoring rule. During our search for the best scoring two component rule, we only consider two component rules in which adding either component has a significant effect. Determining if either component has a significant effect can be done through two hypothesis tests. In the first hypothesis test, we use Fisher's exact test to determine the score of adding $C_2 = V_2$ to the one component rule $C_1 = V_1$. Similarly, in the second hypothesis test, we use Fisher's exact test to score the addition of the component $C_1 = V_1$ to $C_2 = V_2$. The 2-by-2 contingency tables used by the two hypothesis tests are shown in Table 3.

Records from Today with $C_1 = V_1$ and $C_2 = V_2$	Records from Other with $C_1 = V_1$ and $C_2 = V_2$
Records from Today with $C_1 \neq V_1$ and $C_2 = V_2$	Records from Other with $C_1 \neq V_1$ and $C_2 = V_2$

Records from Today with $C_1 = V_1$ and $C_2 = V_2$	Records from Other with $C_1 = V_1$ and $C_2 = V_2$
Records from Today with $C_1 = V_1$ and $C_2 \neq V_2$	Records from Other with $C_1 = V_1$ and $C_2 \neq V_2$

Table 3: 2x2 Contingency Tables for a Two Component Rule

Once we have the scores for both tables, we need to determine if they are significant or not. A score is considered significant if the result of a hypothesis test is significant at the $\alpha = 0.05$ level. If the scores for the two tables are both significant, then the presence of both components has an effect. As a result, the best rule overall for day i is BR_i^2 . On the other hand, if any one of the scores is not significant, then the best rule overall for day i is BR_i^1 .

2.5 n Component Rules

Let BR_i^{k-1} be the best $k - 1$ component rule found for day i . In the general case of finding the best n component rule, the procedure is analogous to that of the previous section. Given BR_i^{k-1} , we produce BR_i^k by greedily adding on the best component, which is found by evaluating all possible attribute-

value pairs as the next component, excluding those already present in components of BR_i^{k-1} . Starting with BR_i^1 , we repeat this procedure until we reach BR_i^n .

In order to determine if the addition of a component is significant, we should in theory test all possible combinations of the n components. In general, we need $2 \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{i}$ such tests. Having this many tests is clearly computationally intensive as n increases. As an approximation, we resort to testing if adding the n th component is significant with respect to the $n - 1$ other components. The two significance tests are as shown in Table 4, where $C_n = V_n$ refers to the last component added and $C_1 = V_1, \dots, C_{n-1} = V_{n-1}$ refers to the conjunction of the previous $n - 1$ components. As before, if both of the Fisher's exact tests return a score less than $\alpha = 0.05$, then we consider the addition of the rule component significant. Due to this step, the probability of having a rule with many components is low because for each component added, it needs to be significant at the 95% level for both of the Fisher's exact tests.

Records from Today with $C_1 = V_1, \dots, C_{n-1} = V_{n-1}$ and $C_n = V_n$	Records from Other with $C_1 = V_1, \dots, C_{n-1} = V_{n-1}$ and $C_n = V_n$
Records from Today with $C_1 = V_1, \dots, C_{n-1} = V_{n-1}$ and $C_n \neq V_n$	Records from Other with $C_1 = V_1, \dots, C_{n-1} = V_{n-1}$ and $C_n \neq V_n$

Records from Today with $C_1 = V_1, \dots, C_{n-1} = V_{n-1}$ and $C_n = V_n$	Records from Other with $C_1 = V_1, \dots, C_{n-1} = V_{n-1}$ and $C_n = V_n$
Records from Today with $\neg(C_1 = V_1, \dots, C_{n-1} = V_{n-1})$ and $C_n = V_n$	Records from Other with $\neg(C_1 = V_1, \dots, C_{n-1} = V_{n-1})$ and $C_n = V_n$

Table 4: 2x2 Contingency Tables for an N Component Rule

2.6 Finding the p-value for a Rule

The algorithm above for determining scores is prone to overfitting due to multiple hypothesis testing. Even if data were generated randomly, most single rules would have insignificant p-values but the best rule would be significant if we had searched over 1000 possible rules. In order to illustrate this point, suppose we follow the standard practice of rejecting the null hypothesis when the p-value is $< \alpha$, where $\alpha = 0.05$. In the case of a single hypothesis test, the probability of a false positive under the null hypothesis would be α , which equals 0.05. On the other hand, if we perform 1000 hypothesis tests, one for each possible rule under consideration, then the probability of a false positive could be as bad as $1 - (1 - 0.05)^{1000} \approx 1$, which is much greater than 0.05 (Miller et al., 2001). Thus, if our algorithm returns a significant p-value, we cannot accept it at face value without adding an adjustment for the multiple hypothesis tests we performed. This problem can be addressed using a Bonferroni correction (Bonferroni, 1936) but this approach would be unnecessarily conservative. Instead, we use a randomization test. Under the null hypothesis of this randomization test, the date and the other ED case attributes are assumed to be independent. Consequently, the case attributes in the data set DB_i remain the same for each record but the date field is shuffled between records from the current day and records from five to eight weeks ago. The full method for the randomization test is shown below.

Let $UCP_i =$ Uncompensated p-value i.e. the score as defined above.

For $j = 1$ to 1000

Let $DB_i^{(j)}$ = newly randomized data set

Let $BR_i^{(j)}$ = Best rule on $DB_i^{(j)}$

Let $UCP_i^{(j)}$ = Uncompensated p-value of $BR_i^{(j)}$ on $DB_i^{(j)}$

Let the compensated p-value of BR_i be CPV_i i.e.

$$CPV_i = \frac{\# \text{ of Randomized Tests in which } UCP_i^{(j)} < UCP_i}{\# \text{ of Randomized Tests}}$$

CPV_i is an estimate of the chance that we would have seen an uncompensated p-value as good as UCP_i if in fact there was no relationship between date and case attributes. Note that we do not use the uncompensated p-value UCP_i after the randomization test. Instead, the compensated p-value CPV_i is used to decide if an alarm should be raised.

The bottleneck in the entire WSARE procedure is the randomization test. If implemented naively, it can be extremely computationally intense. In order to illustrate its complexity, suppose there are M attributes and each attribute can take on K possible values. In addition, let there be N_T records for today and N_B records for the baseline period. Note that typically, N_T is 4 to 20 times smaller than N_B . At iteration j of the randomization test, we need to search for the best scoring rule over $DB_i^{(j)}$. Assuming we limit the number of components in a rule to be two, searching for the best rule using a greedy search requires scoring $KM + K(M - 1)$ rules. Scoring a rule requires us to obtain the entries for the two by two contingency table by counting over $N_T + N_B$ records. Thus, each iteration of the randomization test has a complexity of $(KM + K(M - 1)) * (N_T + N_B)$. With Q iterations, the overall complexity of the randomization test is $O(QKM(N_T + N_B))$.

One of the key optimizations to speeding up the randomization test is the technique of “racing” (Maron and Moore, 1997). If BR_i is highly significant, we run the full 1000 iterations but we stop early if we can show with very high confidence that CPV_i is going to be greater than 0.1. As an example, suppose we have gone through j iterations and let CPV_i^j be the value of CPV_i on the current iteration j (CPV_i^j is calculated as the number of times so far that the best scoring rule on the randomized data set has a lower p-value than the best scoring rule over the original unrandomized data set). Using a normality assumption on the distribution of CPV_i , we can estimate the standard deviation σ_{CPV_i} and form a 95% confidence interval on the true value of CPV_i . This is achieved using the interval $CPV_i^j \pm \frac{1.96\sigma_{CPV_i}}{\sqrt{j}}$. If the lower half of this interval, namely $CPV_i^j - \frac{1.96\sigma_{CPV_i}}{\sqrt{j}}$, is greater than, say 0.1, we are 95% sure that this score will be insignificant at the 0.1 level. On a typical data set where an outbreak is unlikely, the majority of days will result in insignificant p-values. As a result, we expect the racing optimization to allow us to stop early on many days.

2.7 Using FDR to Determine Which p-values are Significant

This algorithm can be used on a day-to-day basis or it can operate over a history of several days to report all significantly anomalous patterns. When using our algorithm on a day-to-day basis, the compensated p-value CPV_i obtained for the current day through the randomization tests can be interpreted at face value. However, when analyzing historical data, we need to characterize the false discovery rate over the group of days in the history, which requires comparing the CPV_i values for each day. Comparison of multiple CPV_i values in the historical window results in a

second overfitting opportunity analogous to that caused by performing multiple hypothesis tests to determine the best rule for a particular day. As an illustration, suppose we took 500 days of randomly generated data. Then, approximately 5 days would have a CPV_i value less than 0.01 and these days would naively be interpreted as being significant. Two approaches can be used to correct this problem. The Bonferroni method (Bonferroni, 1936) aims to reduce the probability of making one or more false positives to be no greater than α . However, this tight control over the number of false positives causes many real discoveries to be missed. The other alternative is Benjamini and Hochberg's False Discovery Rate method, (Benjamini and Hochberg, 1995), which we will refer to as BH-FDR. BH-FDR guarantees that the false discovery rate, which is the expected fraction of the number of false positives over the number of tests in which the null hypothesis is rejected, will be no greater than α_{FDR} . The FDR method is more desirable as it has a higher power than the Bonferroni correction while keeping a reasonable control over the false discovery rate. We incorporate the BH-FDR method into our rule-learning algorithm by first providing an α_{FDR} value and then using BH-FDR to find the cutoff threshold for determining which p-values are significant.

3. WSARE 3.0

Many detection algorithms (Goldenberg et al., 2002; Zhang et al., 2003; Fawcett and Provost, 1997) assume that the observed data consist of cases from background activity, which we will refer to as the baseline, plus any cases from irregular behavior. Under this assumption, detection algorithms operate by subtracting away the baseline from recent data and raising an alarm if the deviations from the baseline are significant. The challenge facing all such systems is to estimate the baseline distribution using data from historical data. In general, determining this distribution is extremely difficult due to the different trends present in surveillance data. Seasonal variations in weather and temperature can dramatically alter the distribution of surveillance data. For example, flu season typically occurs during mid-winter, resulting in an increase in ED cases involving respiratory problems. Disease outbreak detectors intended to detect epidemics such as SARS, West Nile Virus and anthrax are not interested in detecting the onset of flu season and would be thrown off by it. Day of week variations make up another periodic trend. Figure 3, which is taken from Goldenberg et al. (2002), clearly shows the periodic elements in cough syrup and liquid decongestant sales.

Choosing the wrong baseline distribution can have dire consequences for an early detection system. Consider once again a database of ED records. Suppose we are presently in the middle of flu season and our goal is to detect anthrax, not an influenza outbreak. Anthrax initially causes symptoms similar to those of influenza. If we choose the baseline distribution to be outside of the current flu season, then a comparison with recent data will trigger many false anthrax alerts due to the flu cases. Conversely, suppose we are not in the middle of flu season and that we obtain the baseline distribution from the previous year's influenza outbreak. The system would now consider high counts of flu-like symptoms to be normal. If an anthrax attack occurs, it would be detected late, if at all.

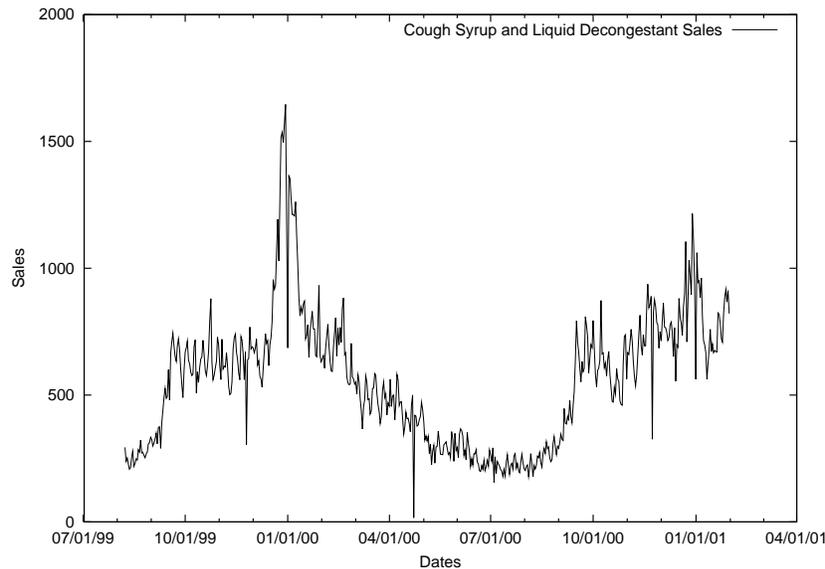


Figure 3: Cough syrup and liquid decongestant sales from (Goldenberg et al., 2003)

There are clearly tradeoffs when defining this baseline distribution. At one extreme, we would like to capture any current trends in the data. One solution would be to use only the most recent data, such as data from the previous day. This approach, however, places too much weight on outliers that may only occur in a short but recent time period. On the other hand, we would like the baseline to be accurate and robust against outliers. We could use data from all previous years to establish the baseline. This choice would smooth out trends in the data and likely raise alarms for events that are due to periodic trends.

In WSARE 2.0, we made the baseline distribution to be raw data obtained from selected historical days. For example, we chose data from 35, 42, 49, and 56 days prior to the current day under examination. These dates were chosen to incorporate enough data so that seasonal trends could be captured and they were also chosen to avoid weekend versus weekday effects by making all comparisons from the same day of week. This baseline was chosen manually in order to tune the performance of WSARE 2.0 on the data set. Ideally, the detection system should determine the baseline automatically.

In this section, we describe how we use a Bayesian network to represent the joint probability distribution of the baseline. From this joint distribution, we represent the baseline distributions from the conditional distributions formed by conditioning on what we term *environmental attributes*. These attributes are precisely those attributes that account for trends in the data, such as the season, the current flu level and the day of week.

3.1 Creating the Baseline Distribution

Learning the baseline distribution involves taking all records prior to the past 24 hours and building a Bayesian network from this subset. During the structure learning, we differentiate between *environmental attributes*, which are attributes that cause trends in the data, and *response attributes*, which are the remaining attributes. The environmental attributes are specified by the user based on the user's knowledge of the problem domain. If there are any latent environmental attributes

that are not accounted for in this model, the detection algorithm may have some difficulties. However, as will be described later on in Section 4, WSARE 3.0 was able to overcome some hidden environmental attributes in our simulator.

While learning the structure of the Bayesian network, environmental attributes are prevented from having parents because we are not interested in predicting their distributions, but rather, we want to use them to predict the distributions of the response attributes. In general, any structure learning algorithm can be used in this step as long as it follows this restriction. In fact, the structure search can even exploit this constraint by avoiding search paths that assign parents to the environmental attributes.

We experimented with using hillclimbing to learn the Bayesian network structure and found it to be both slow and prone to being trapped in local optima. As a result, we developed an efficient structure search algorithm called Optimal Reinsertion based on ADTrees (Moore and Lee, 1998). Unlike hillclimbing, which performs a single modification to a directed acyclic graph (DAG) on each step, Optimal Reinsertion is a larger scale search operator that is much less prone to local optima. Optimal Reinsertion first picks a target node T from the DAG, disconnects T from the graph, and efficiently finds the optimal way to reinsert T back into the graph according to the scoring function. The details of this algorithm can be found in (Moore and Wong, 2003).

We have often referred to environmental attributes as attributes that cause periodic trends. Environmental attributes, however, can also include any source of information that accounts for recent changes in the data. For example, suppose we detect that a botulism outbreak has occurred and we would still like to be on alert for any anthrax releases. We can add “Botulism Outbreak” as an environmental attribute to the network and supplement the current data with information about the botulism outbreak. Incorporating such knowledge into the Bayesian network allows WSARE to treat events due to the botulism outbreak as part of the baseline.

Once the Bayesian network is learned, we have a joint probability distribution for the data. We would like to produce a conditional probability distribution, which is formed by conditioning on the values of the environmental attributes. Suppose that today is February 21, 2003. If the environmental attributes were *Season* and *Day of Week*, we would set $Season = Winter$ and $Day of Week = Weekday$. Let the response attributes in this example be X_1, \dots, X_n . We can then obtain the probability distribution $P(X_1, \dots, X_n \mid Season = Winter, Day of Week = Weekday)$ from the Bayesian network. For simplicity, we represent the conditional distribution as a data set formed by sampling a large number of records from the Bayesian network conditioned on the environmental attributes. The number of samples is specified by the parameter *num_baseline_samples*, which has to be large enough to ensure that samples with rare combinations of attributes will be present. In general, this number will depend on the learned Bayesian network’s structure and the parameters of the network. We chose to sample 10000 records because we determined empirically that this number is a reasonable compromise between running time and accuracy on our data. We will refer to this sampled data set as $DB_{baseline}$. The data set corresponding to the records from the past 24 hours of the current day will be named DB_{recent} .

We used a sampled data set instead of using inference mainly for simplicity. Inference might be faster than sampling to obtain the conditional probability $P(X_1, \dots, X_n \mid \text{Environmental Attributes})$, especially when the learned Bayesian networks are simple. However, if inference is used, it is somewhat unclear how to perform the randomization test. With sampling, on the other hand, we only need to generate $DB_{baseline}$ once and then we can use it for the randomization test to obtain the p-values for all the rules. In addition, sampling is easily done in an efficient manner since

environmental attributes have no parents. While a sampled data set provides the simplest way of obtaining the conditional distribution, we have not completely ignored the possibility of using inference to speed up this process. We would like to investigate this direction further in our future work.

3.2 Dealing with New Hospitals Coming Online

WSARE 3.0 assumes that the baseline distribution remains relatively stable, with the environmental attributes accounting for the only sources of variation. However, in a real life situation where data are pooled from various EDs around a city, new hospitals frequently come online and become a new source of data to be monitored. These new data sources cause a shift from the baseline distribution that is not accounted for in WSARE 3.0. For example, suppose a children's hospital begins sending data to the surveillance system. In this case, WSARE 3.0 would initially detect an anomalous pattern due to an increase in the number of cases involving children from the part of the city where the children's hospital is located. Over time, WSARE 3.0 would eventually incorporate the newly added hospital's data into its baseline.

In general, this problem of a shifted distribution is difficult to address. We approach this issue by ignoring the new data sources until we have enough data from them to incorporate them into the baseline. Our solution relies on the data containing an attribute such as *Hospital ID* that can identify the hospital that the case originated from. HIPAA regulations can sometimes prevent ED data from containing such identifying attributes. In this case, we recommend using WSARE 2.0 with a recent enough baseline period in order to avoid instabilities due to new data sources. Whenever the data includes a *Hospital ID* attribute, we first build a list of hospitals that provide data for the current day. For each hospital in this list, we keep track of the first date a case came from that particular hospital. If the current day is less than a year after the first case date, we consider that hospital to have insufficient historical data for the baseline and we ignore all records from that hospital. For each hospital with sufficient historical records, we then build a Bayesian network using only historical data originating from that particular hospital.

In order to produce the baseline data set, we sample a total of 10000 records from all the hospital Bayesian networks. Let hospital h have n_h records on the current day and suppose there are H hospitals with sufficient historical data for the current date. Then let $N_h = \sum_{h=1}^H n_h$. Each hospital Bayesian network contributes $10000 * \frac{n_h}{N_h}$ number of samples to the baseline data set. As an example, suppose we have 5 hospitals with 100 records each. Furthermore, assume that we can ignore the fourth hospital's records since its first case is less than a year prior to the current date. We are then left with 4 hospitals with 100 records each. After we build the Bayesian network for each hospital, we sample 2500 records from the Bayesian network belonging to each of the four hospitals.

4. Evaluation

Validation of early outbreak detection algorithms is generally a difficult task due to the type of data required. Health-care data during a known disease outbreak, either natural or induced by a bio-agent release, are extremely limited. Even if such data were plentiful, evaluation of biosurveillance algorithms would require the outbreak periods in the data to be clearly labelled. This task requires an expert to inspect the data manually, making this process extremely slow. Consequently, such labelled data would still be scarce and making statistically significant conclusions with the results of detection algorithms would be difficult. Furthermore, even if a group of epidemiologists were to

be assembled to label the data, there would still be disagreements as to when an outbreak begins and ends.

As a result of these limitations, we validate the WSARE algorithms on data from a simulator which we will refer to as the city Bayesian network (CityBN) simulator. The CityBN simulator is based on a large Bayesian network that introduces temporal fluctuations based on a variety of factors. The structure and the parameters for this Bayesian network are created by hand. This simulator is not intended to be a realistic epidemiological model. Instead, the model is designed to produce extremely noisy data sets that are a challenge for any detection algorithm. In addition to simulated data, we also include WSARE output from ED data from an actual city. Due to the fact that epidemiologists have not analyzed this real world data set for known outbreaks, we are only able to provide annotated results from the runs of WSARE.

4.1 The CityBN Simulator

The city in the CityBN simulator consists of nine regions, each of which contains a different sized population, ranging from 100 people in the smallest area to 600 people in the largest section, as shown in Table 5. We run the simulation for a two year period starting from January 1, 2002 to December 31, 2003. The environment of the city is not static, with weather, flu levels and food conditions in the city changing from day to day. Flu levels are typically low in the spring and summer but start to climb during the fall. We make flu season strike in winter, resulting in the highest flu levels during the year. Weather, which only takes on the values of hot or cold, is as expected for the four seasons, with the additional feature that it has a good chance of remaining the same as it was yesterday. Each region has a food condition of good or bad. A bad food condition facilitates the outbreak of food poisoning in the area.

NW (100)	N (400)	NE (500)
W (100)	C (200)	E (300)
SW (200)	S (200)	SE (600)

Table 5: The geographic regions in the CityBN simulator with their populations in parentheses

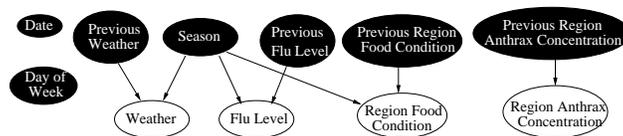


Figure 4: City Status Bayesian Network

We implement this city simulation using a single large Bayesian network. For simplicity, we will describe this large Bayesian network in two parts, as shown in Figures 4 and 5. The subnetwork shown in Figure 4 is used to create the state of the city for a given day. Given the state of the city, the network in Figure 5 is used to generate records for individual patients.

We use the convention that any nodes shaded black in the subnetwork are set by the system and do not have their values generated probabilistically. Due to space limitations, instead of showing

eighteen separate nodes for the current and previous food conditions of each region in Figure 4, we summarize them using the generic nodes *Region Food Condition* and *Previous Region Food Condition* respectively. This same space saving technique is used for the current and previous region anthrax concentrations. Most of the nodes in this subnetwork take on two to three values. For each day, after the black nodes have their values set, the values for the white nodes are sampled from the subnetwork. These records are stored in the City Status (CS) data set. The simulated anthrax release is selected for a random date during a specified time period. One of the nine regions is chosen randomly for the location of the simulated release. On the date of the release, the *Region Anthrax Concentration* node is set to have the value of *High*. The anthrax concentration remains high for the affected region for each subsequent day with an 80% probability. This probability is chosen in order to ensure that enough individuals in the simulation are being infected by anthrax over an extended period of time after the attack.

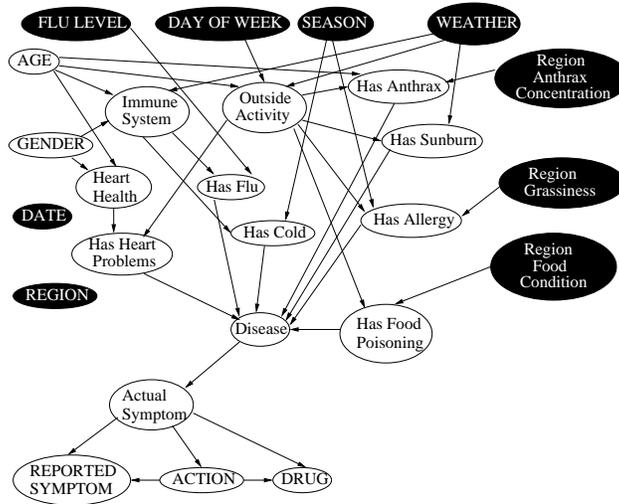


Figure 5: Patient Status Bayesian Network

Table 6: Examples of two records in the PS data set

Location	NW	N
Age	Child	Senior
Gender	Female	Male
Flu Level	High	None
Day of Week	Weekday	Weekday
Weather	Cold	Hot
Season	Winter	Summer
Action	Absent	ED visit
Reported Symptom	Nausea	Rash
Drug	None	None
Date	Jan-01-2002	Jun-21-2002

The second subnetwork used in our simulation produces individual health care cases. Figure 5 depicts the Patient Status (PS) network. On each day, for each person in each region, we sample

the individual's values from this subnetwork. The black nodes first have their values assigned from the CS data set record for the current day. For the very first day, the black nodes are assigned a set of initial values. The white nodes are then sampled from the PS network. Each individual's health profile for the day is thus generated. The nodes *Flu Level*, *Day of Week*, *Season*, *Weather*, *Region Grassiness*, and *Region Food Condition* are intended to represent environmental variables that affect the upswings and downswings of a disease. The *Region Grassiness* nodes indicate the amount of pollen in the air and thus affect the allergies of a patient. We choose these environmental variables because they are the most common factors influencing the health of a population. Two of the environmental variables, namely *Region Grassiness* and *Region Food Condition*, are hidden from the detection algorithm while the remaining environmental attributes are observed. We choose to hide these two attributes because the remaining four attributes that are observed are typically considered when trying to account for temporal trends in biosurveillance data.

As for the other nodes, the *Disease* node indicates the status of each person in the simulation. We assume that a person is either healthy or they can have, in order of precedence, allergies, the cold, sunburn, the flu, food poisoning, heart problems or anthrax. If the values of the parents of the *Disease* node indicate that the individual has more than one disease, the *Disease* node picks the disease with the highest precedence. This simplification prevents individuals from having multiple diseases. A sick individual then exhibits one of the following symptoms: none, respiratory problems, nausea, or a rash. Note that in our simulation, as in real life, different diseases can exhibit the same symptoms, such as a person with the flu can exhibit respiratory problems as could a person with anthrax. The actual symptom associated with a person may not necessarily be the same as the symptom that is reported to health officials. Actions available to a sick person included doing nothing, buying medication, going to the ED, or being absent from work or school. As with the CS network, the arities for each node in the PS network are small, ranging from two to four values. If the patient performs any action other than doing nothing, the patient's health care case is added to the PS data set. Only the attributes in Figure 5 labelled with uppercase letters are recorded, resulting in a great deal of information being hidden from the detection algorithm, including some latent environmental attributes. The number of cases the PS network generates daily is typically in the range of 30 to 50 records. Table 6 contains two examples of records in the PS data set.

We run six detection algorithms on 100 different PS data sets. Each data set is generated for a two year period, beginning on January 1, 2002 and ending December 31, 2003. The detection algorithms train on data from the first year until the day being monitored while the second year is used for evaluation. The anthrax release is randomly chosen in the period between January 1, 2003 and December 31, 2003.

We try to simulate anthrax attacks that are not trivially detectable. Figure 6 plots the total count of health-care cases on each day during the evaluation period while Figure 7 plots the total count of health-care cases involving respiratory symptoms for the same simulated data set. A naive detection algorithm would assume that the highest peak in this graph would be the date of the anthrax release. However, the anthrax release occurs on day index 74,409, which is clearly not the highest peak in either graph. Occasionally the anthrax releases affects such a limited number of people that it is undetected by all the algorithms. Consequently, we only use data sets with more than eight reported anthrax cases on any day during the attack period.

The following paragraphs describe the six detection algorithms that we run on the data sets. Three of these methods, namely the control chart, moving average, and ANOVA regression algorithms, operate on univariate data. We apply these three algorithms to two different univariate data

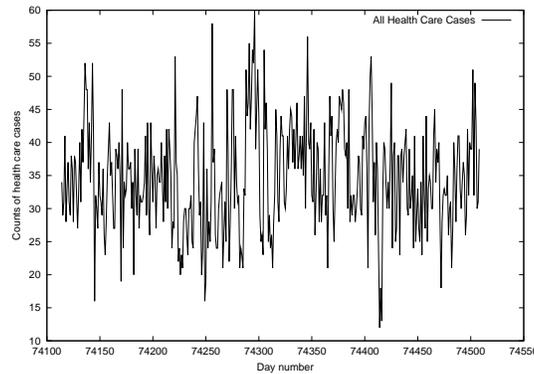


Figure 6: Daily counts of health-care data

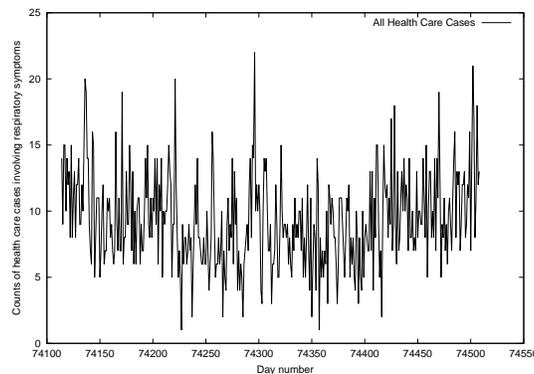


Figure 7: Daily counts of health-care data involving respiratory symptoms

sets – one data set is composed of total daily counts and the other of daily counts of cases involving respiratory symptoms. The remaining three algorithms are variations on WSARE.

The Control Chart Algorithm The first algorithm used is a common anomaly detection algorithm called a control chart. This detector determines the mean and variance of the total number of records on each day in the PS data set during the training period. A threshold is calculated based on the formula below, in which Φ^{-1} is the inverse to the cumulative distribution function of a standard normal while the p-value is supplied by the user.

$$\text{threshold} = \mu + \sigma * \Phi^{-1}\left(1 - \frac{\text{p-value}}{2}\right)$$

If the aggregate daily counts of health care data exceeds this threshold during the evaluation period, the control chart raises an alarm. We use a training period of January 1, 2002 to December 31, 2002.

Moving Average Algorithm The second algorithm that we use is a moving average algorithm that predicts the count for the current day as the average of counts from the previous 7 days. The window of 7 days is intended to capture any recent trends that might appear in the data. An alarm level is generated by fitting a Gaussian to data prior to the current day and obtaining a p-value for

the current day's count. The mean and standard deviation for the Gaussian is calculated using data from 7 days before the current day.

ANOVA Regression A simple detector that accounts for environmental factors is ANOVA regression, which is simply linear regression supplemented with covariates for the environmental variables. We include 6 covariates for the days of the week, 3 for the seasons and one for the daily aggregate count from the previous day. ANOVA regression is a fairly powerful detector when temporal trends are present in the data, as was shown in (Buckeridge et al., 2005).

WSARE 2.0 WSARE 2.0 is also evaluated, using a baseline distribution of records from 35, 42, 49 and 56 days before the current day. The attributes used by WSARE 3.0 as environmental attributes are ignored by WSARE 2.0. If these attributes are not ignored, WSARE 2.0 would report many trivial anomalies. For instance, suppose that the current day is the first day of fall, making the environmental attribute *Season = Fall*. Furthermore, suppose that the baseline is taken from the summer season. If the environmental attributes are not ignored, WSARE 2.0 would notice that 100% of the records for the current day have *Season = Fall* while 0% of the records in the baseline data set match this rule.

WSARE 2.5 Instead of building a Bayesian network over the past data, WSARE 2.5 simply builds a baseline from all records prior to the current period with their environmental attributes equal to the current day's. In our simulator, we use the environmental attributes *Flu Level*, *Season*, *Day of Week* and *Weather*. To clarify this algorithm, suppose for the current day we have the following values of these environmental attributes: *Flu Level = High*, *Season = Winter*, *Day of Week = Weekday* and *Weather = Cold*. Then $DB_{baseline}$ would contain only records before the current period with environmental attributes having exactly these values. It is possible that no such records exist in the past with exactly this combination of environmental attributes. If there are fewer than five records in the past that matched, WSARE 2.5 can not make an informed decision when comparing the current day to the baseline and simply reports nothing for the current day.

WSARE 3.0 WSARE 3.0 uses the same environmental attributes as WSARE 2.5 but builds a Bayesian network for all data from January 1, 2002 to the day being monitored. We hypothesize that WSARE 3.0 would detect the simulated anthrax outbreak sooner than WSARE 2.5 because 3.0 can handle the cases where there are no records corresponding to the current day's combination of environmental attributes. The Bayesian network is able to generalize from days that do not match today precisely, producing an estimate of the desired conditional distribution. For efficiency reasons, we allow WSARE 3.0 to learn the network structure from scratch once every 30 days on all data since January 1, 2002. On intermediate days, WSARE 3.0 simply updates the parameters of the previously learned network without altering its structure. In practice, we expect WSARE 3.0 to be used in this way since learning the network structure on every day may be very expensive computationally.

4.1.1 RESULTS

In order to evaluate the performance of the algorithms, we plot an Activity Monitoring Operating Characteristic (AMOC) curve (Fawcett and Provost, 1999), which is similar to an ROC curve. On the AMOC curves to follow, the x-axis indicates the number of false positives per month while the y-axis measures the detection time in days. For a given alarm threshold, we plot the performance of the algorithm at a particular false positive level and detection time on the graph. As an example,

suppose we are dealing with an alarm threshold of 0.05. We then take all the alarms generated by an algorithm, say WSARE 3.0, that have a p-value less than or equal to 0.05. Suppose there are two such alarms, with one alarm appearing 5 days before the simulated anthrax release, which would be considered a false positive, and the other appearing 3 days after the release, making the detection time 3 days. If we run the detection algorithms for 1 month, then we would plot a point at (1,3).

We then vary the alarm threshold in the range of 0 to 0.2 and plot points at each threshold value. For a very sensitive alarm threshold such as 0.2, we expect a higher number of false positives but a lower detection time. Hence the points corresponding to a sensitive threshold would be on the lower right hand side of the graph. Conversely, an insensitive alarm threshold like 0.01 would result in a lower number of false positives and a higher detection time. The corresponding points would appear on the upper left corner of the graph.

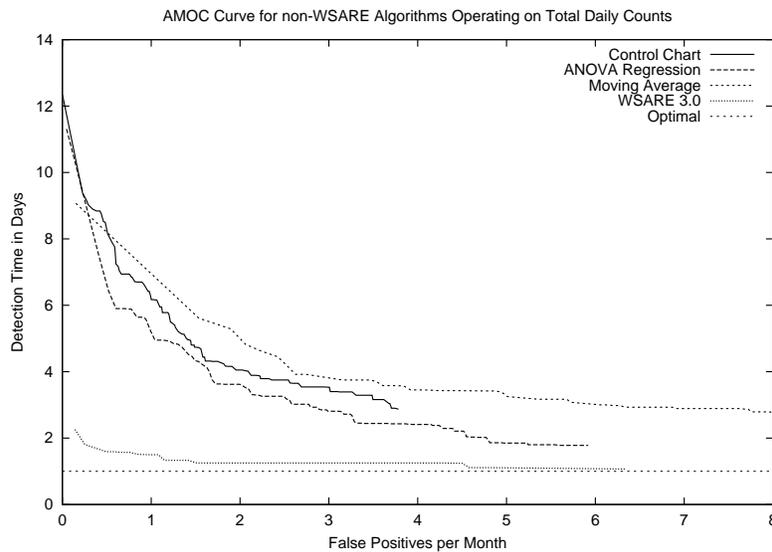


Figure 8: AMOC curves comparing WSARE 3.0 to univariate algorithms operating on total daily counts from the CityBN simulator

Figures 8 to 10 plot the AMOC curve, averaged over the 100 data sets, with an alarm threshold increment of 0.001. On these curves, the optimal detection time is one day, as shown by the dotted line at the bottom of the graph. We add a one day delay to all detection times to simulate reality where current data is only available after a 24 hour delay. Any alert occurring before the start of the simulated anthrax attack is treated as a false positive. Detection time is calculated as the first alert raised after the release date. If no alerts are raised after the release, the detection time is set to 14 days.

Figures 8 and 9 show that WSARE 3.0 clearly outperform the univariate algorithms when the univariate algorithms operate on the total daily counts and also when the univariate algorithms operate on the daily counts of cases involving respiratory symptoms. In Figure 10, WSARE 2.5 and WSARE 3.0 outperform the other algorithms in terms of the detection time and false positive tradeoff. For a false positive rate of one per month, WSARE 2.5 and WSARE 3.0 are able to detect the anthrax release within a period of one to two days. The Control Chart, moving average, ANOVA regression and WSARE 2.0 algorithms are thrown off by the periodic trends present in the PS data.

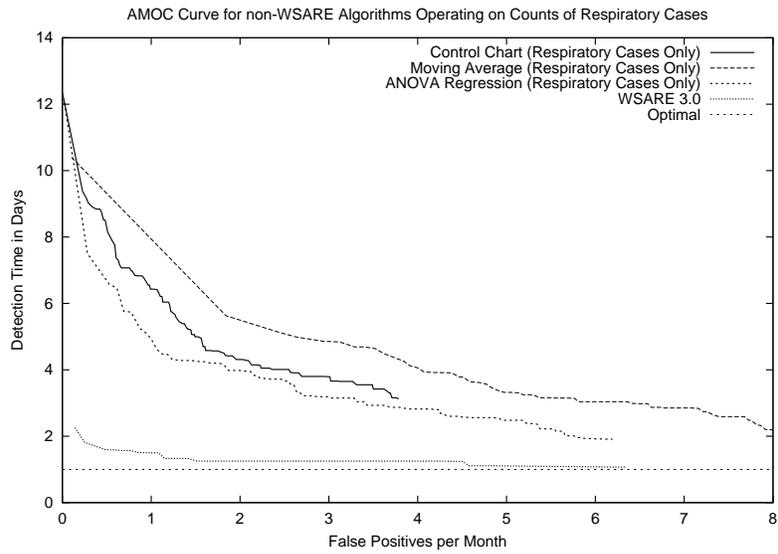


Figure 9: AMOC curves comparing WSARE 3.0 to univariate algorithms operating on cases involving respiratory symptoms from the CityBN simulator

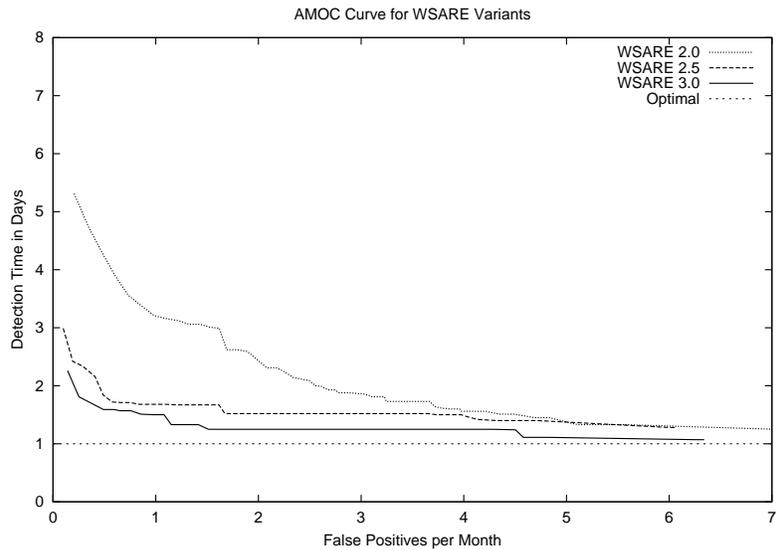


Figure 10: AMOC curves for WSARE variants operating on CityBN data

We previously proposed that WSARE 3.0 would have a better detection time than WSARE 2.5 due to the Bayesian network’s ability to produce a conditional distribution for a combination of environmental attributes that may not exist in the past data. After checking the simulation results for which WSARE 3.0 outperformed WSARE 2.5, we conclude that in some cases, our proposition is true. In others, the p-values estimated by WSARE 2.5 are not as low as those of version 3.0. The

baseline distribution of WSARE 2.5 is likely not as accurate as the baseline of WSARE 3.0 due to smoothing performed by the Bayesian network. The false positives found by WSARE 2.5 and WSARE 3.0 are likely due to other non-anthrax illnesses that are not accounted for in the Bayesian network. Had we explicitly added a Region Food Condition environmental attribute to the Bayesian network, this additional information would likely have reduced the false positive count.

Figures 11 to 14 illustrate the various outbreak sizes in the simulated data by plotting the number of anthrax cases per day during the outbreak period. Since the outbreak sizes and durations are randomly generated for each of the 100 data sets, we do not have room to show plots for each data set. Instead, we include representative plots of the outbreaks that appeared in our simulated data. Figure 11 represents a large scale outbreak which was easily detected on the first day by most algorithms. Large scale outbreaks were rare in our simulated data. Figure 12 is a representative plot of a medium scale outbreak that is most common in the data. The particular outbreak shown in Figure 12 is also detected by WSARE 3.0 on the first day for an alarm threshold of 0.005. Small scale outbreaks, as shown in Figure 13, are the most difficult to detect. WSARE 3.0 detects the outbreak in Figure 13 on the third day with a very insensitive alarm threshold of 0.005. Figure 14 contains an outbreak that WSARE 3.0 is unable to detect using an alarm threshold of 0.03.

We also conduct four other experiments to determine the effect of varying certain parameters of WSARE 3.0. In the first experiment, we use a Bonferroni correction to correct for multiple hypothesis testing instead of a randomization test. The AMOC curve for the results, as shown in Figure 15 indicate that the Bonferroni correction results are almost identical to those of the randomization test. This similarity was expected because on each day, there are approximately only 50 hypothesis tests being performed to find the best scoring rule and the hypothesis tests are weakly dependent on each other. However, as the number of hypothesis tests increases and as the dependence between the hypothesis tests increases, the results of the randomization test should be better than those of the Bonferroni correction.

In order to illustrate the advantages of the randomization test, we produce dependent hypothesis tests in WSARE by creating attributes that are dependent on each other. We generate a data set using a Markov chain X_0, \dots, X_n in which the states of each random variable in the chain become the attributes in the data set. Each random variable X_t in the Markov chain can be in state A , B , C , or D , except for X_0 which always starts at A . At each time step t , the random variable X_t retains the state of X_{t-1} in the Markov chain with a 90% chance. With a 10% chance, X_t takes on the next state in the ordered sequence A , B , C and D . As an example, if $X_{t-1} = A$, X_t can remain as A or it can become B . If $X_{t-1} = D$, X_t can retain the same state as X_{t-1} or transition back to the state A , which is the first state of the ordered sequence. We use this model to generate 150 days worth of data in which each day contains 1000 records and each record contains 100 attributes. We then sample 14 days of data with the same characteristics except the Markov chain is altered slightly so that each random variable X_t remains in the same state as X_{t-1} with an 89% probability. Thirty data sets, each containing a total of 164 days are produced. Two variations of WSARE 2.0, one with a randomization test and the other with a Bonferroni correction, are applied to these thirty data sets in order to detect the change.

Figure 16 plots the average AMOC curve of this experiment. As the graph illustrates, at a false positive rate of less than 0.4 per month, the randomization test has a much better detection time. Upon further analysis, we find that the reduced performance of the Bonferroni correction are due to a much higher number of false positives. As an example, we find that WSARE often notices that a rule such as $X_{27} = C$ AND $X_{96} = B$ produces a very good score. The Bonferroni correction deals

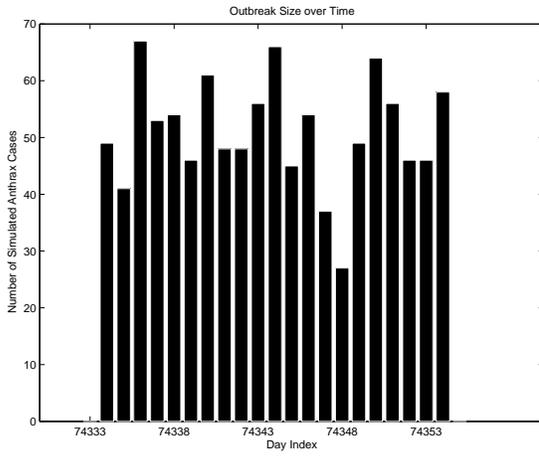


Figure 11: An example of a large scale outbreak in the CityBN data

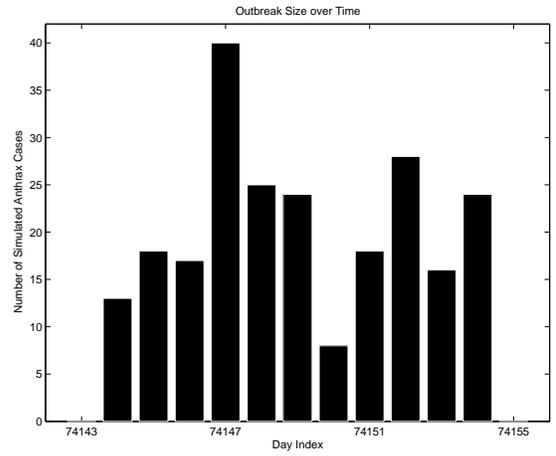


Figure 12: An example of a medium scale outbreak in the CityBN data

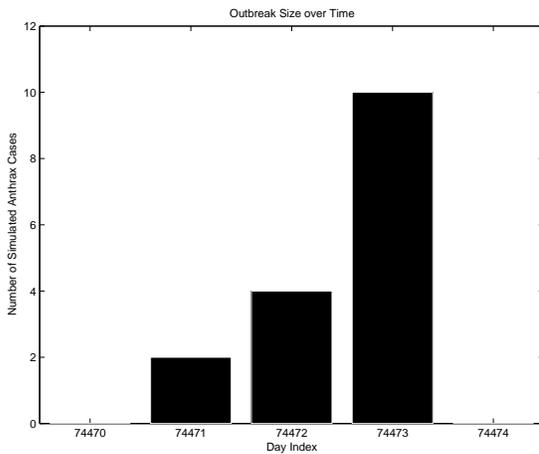


Figure 13: An example of a small scale outbreak in the CityBN data

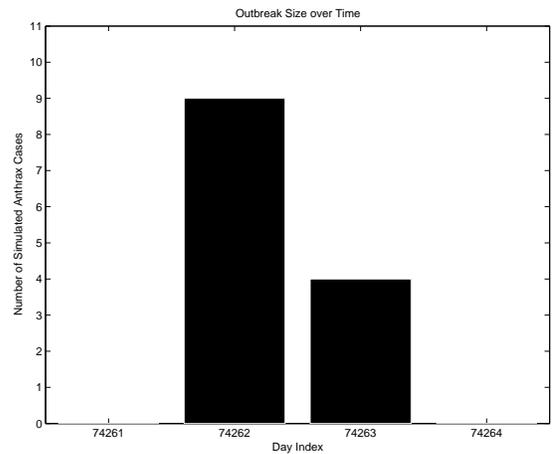


Figure 14: An example of an outbreak that was not detected in the CityBN data by WSARE 3.0 with an alarm threshold of 0.03

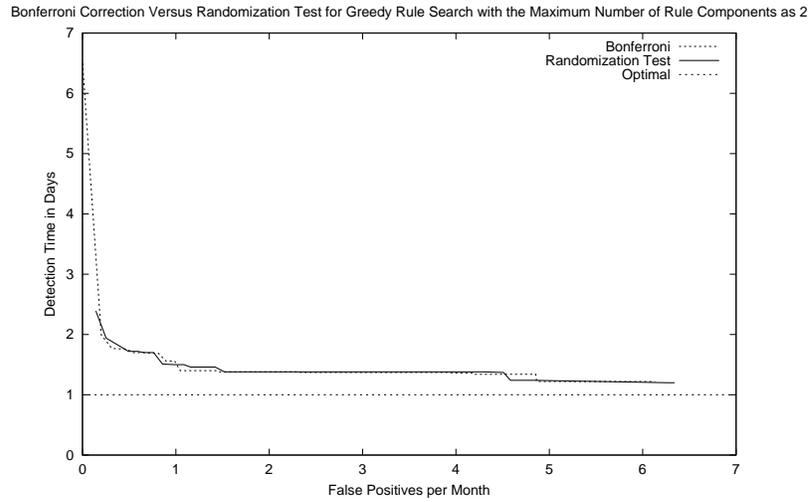


Figure 15: The Bonferroni correction version of WSARE versus the randomization test version on the CityBN data

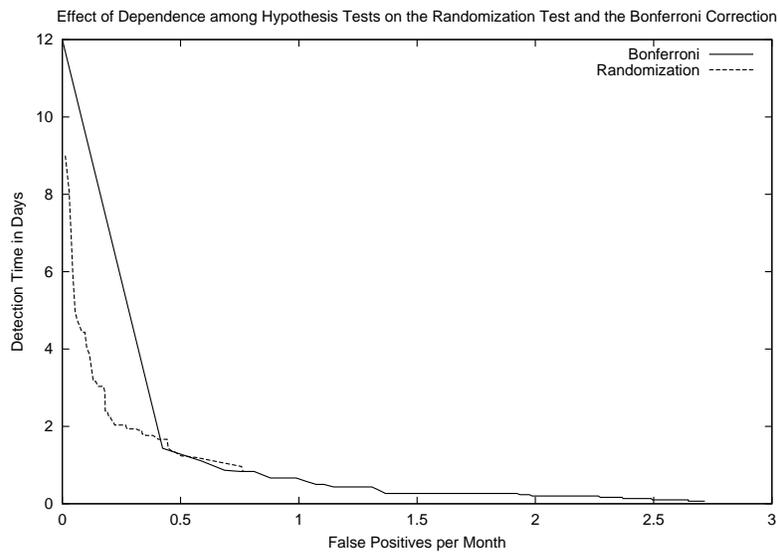


Figure 16: A comparison between the Bonferroni correction version of WSARE and the randomization test version on data generated from a Markov chain

with the multiple hypothesis problem by simply multiplying the score with the number of hypothesis tests. Although there are a high number of hypothesis tests in this experiment, multiplying by the number of hypothesis tests still results in a low compensated p-value. The randomization test, on the other hand, notices that although the score is very good, the probability of finding an equal or better score for another rule, such as $X_{46} = A$ AND $X_{94} = B$ is quite high because of the dependence

between attributes. Thus, the resulting compensated p-value from the randomization test is quite high, signifying that the pattern defined by the rule is not so unusual after all.

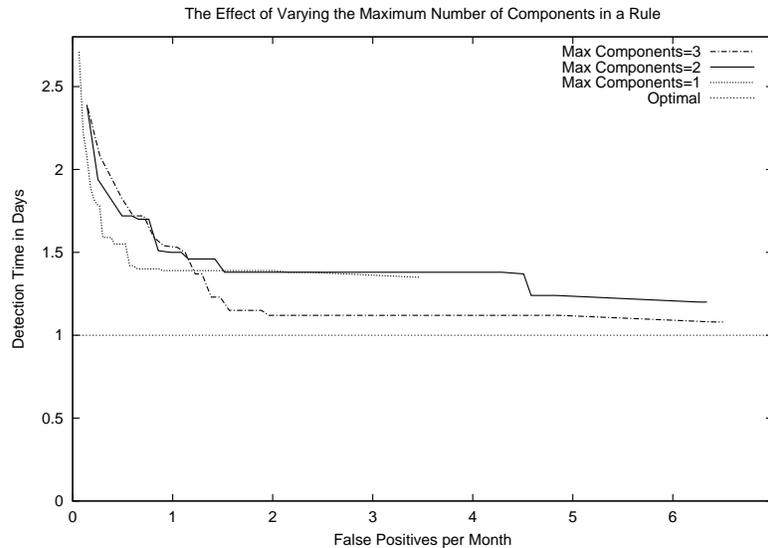


Figure 17: The effect of varying the maximum number of components for a rule on the AMOC curve for CityBN data

The second experiment involves varying the maximum components allowed per rule from one to three. As seen on the AMOC curve in Figure 17, the variations do not seem significantly different to the left of the one false positive per month mark. However, after this point, a version of WSARE with a three component limit outperforms the other two variations. By setting the maximum number of components per rule to be three, WSARE is capable of being more expressive in its description of anomalous patterns. On the other hand, WSARE also guards against overfitting by requiring each component added to be 95% significant for the two hypothesis tests performed in Section 2.5. This criterion makes the addition of a large number of rule components unlikely and we expect the optimal number of components to be about two or three.

The third experiment involves changing the rule search to be exhaustive rather than greedy. Note that if we compare the score of the best rule found by the exhaustive method against that found by the greedy method, the exhaustive method would unquestionably find a rule with an equal or greater score than the greedy method. In Figure 18, however, we compare the performance of the two algorithms using AMOC curves. Each coordinate on the AMOC curve is a result of a compensated p-value produced by the randomization test and not the rule score. Thus, even though an exhaustive rule search will always equal or outperform a greedy rule search in terms of the best rule score, it is not guaranteed to be superior to the greedy rule search on an AMOC curve due to the fact that the randomization test adjusts the rule score for multiple hypothesis testing. In Figure 18, we plot the AMOC curves comparing the average performance for both the exhaustive and greedy algorithms over 100 experiments; we do not show the confidence intervals in order to avoid clutter. The confidence intervals for both the greedy and the exhaustive curves do overlap substantially. Therefore, there appears to be no significant difference between the two algorithms for the data

WHAT'S STRANGE ABOUT RECENT EVENTS

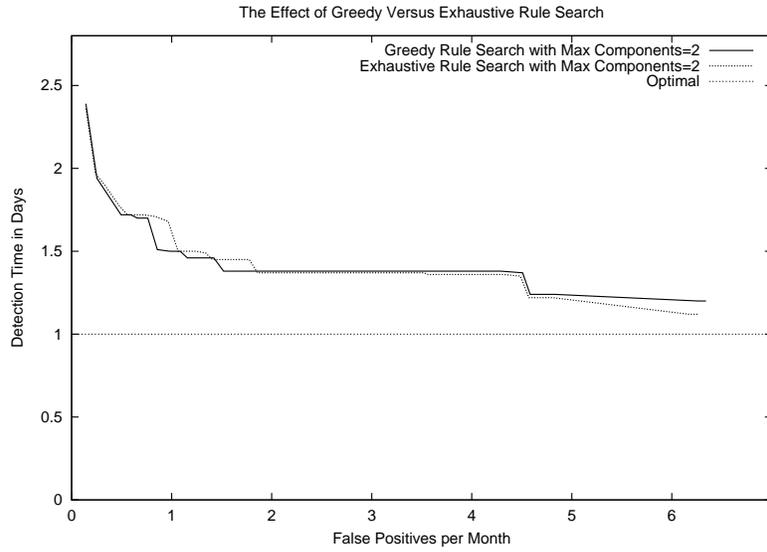


Figure 18: AMOC curves for greedy versus exhaustive rule search for CityBN data

from this simulator. We measure the exhaustive search to be 30 times slower than the greedy search. Since the AMOC curves are nearly identical for our simulated data, we prefer the greedy search.

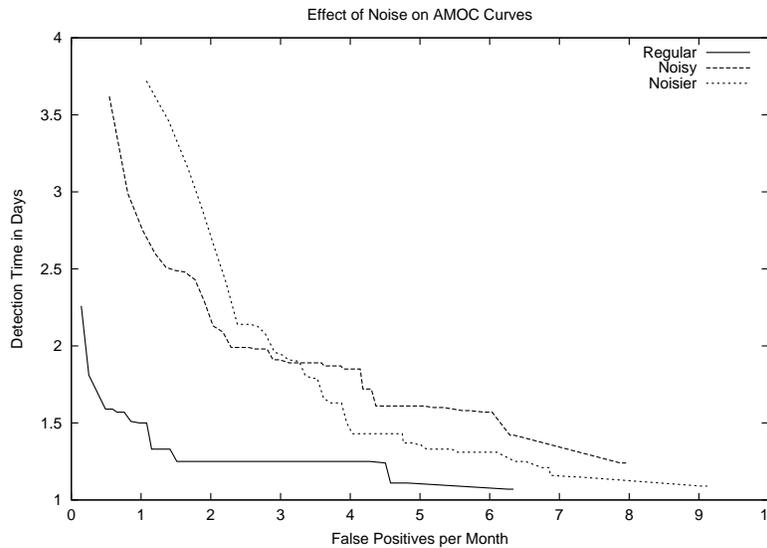


Figure 19: The effect of increased noise levels in the data on WSARE 3.0

Finally, we experiment with adding noise to the data by increasing the number of ED cases due to allergies, food poisoning, sunburns and colds. We increase the noise levels by increasing the probabilities of *Region Food Condition = bad*, *Has Allergy = true*, *Has Cold = true*, and

Has Sunburn = true in their respective conditional probability tables. Note that these nodes are all not visible in the output data. Increasing these probabilities involves changes to many entries of the conditional probability tables and we do not have space to list all of the changes. In general, we increase the probabilities of the corresponding entries in the conditional probability tables by approximately 0.004-0.005. We cannot say specifically how many noisy cases are generated since this amount fluctuates over time.

We produce 100 data sets with increased noise levels which we will refer to as “Noisy” and we also produce another 100 data sets with even more noise which we will refer to as “Noisier”. The “Regular” data sets are the 100 data sets used in all previous experiments. We then apply WSARE 3.0 to these three groups. The average AMOC curve for each group of 100 data sets is plotted in Figure 19. As in previous experiments, we use the environmental attributes of *Flu Level*, *Season*, *Day of Week* and *Weather*. As shown in Figure 19, both the detection time and the false positive rate degrade with increased noise levels.

4.2 Annotated Output of WSARE 3.0 on Actual ED Data for 2001

We also test the performance of WSARE 3.0 on actual ED data from a major US city. This database contains almost seven years worth of data, with personal identifying information excluded in order to protect patient confidentiality. The attributes in this database include date of admission, coded hospital ID, age decile, gender, syndrome information, discretized home latitude, discretized home longitude, discretized work latitude, discretized work longitude and both home location and work location on a coarse latitude-longitude grid. In this data, new hospitals come online and begin submitting data during the time period that the data is collected. We use the solution described in Section 3.2 to address this problem. WSARE operates on data from the year 2001 and is allowed to use over five full years worth of training data from the start of 1996 to the current day. The environmental attributes used are month, day of week and the number of cases from the previous day with respiratory problems. The last environmental attribute is intended to be an approximation to the flu levels in the city. We use a one-sided Fisher’s exact test to score the rules such that only rules corresponding to an upswing in recent data are considered. In addition, we apply the Benjamini-Hochberg FDR procedure with $\alpha_{FDR} = 0.1$.

The following list contains the significant anomalous patterns found in the real ED data for the year 2001.

1. 2001-02-20: SCORE = -2.15432e-07 PVALUE = 0
 15.9774% (85/532) of today’s cases have Viral Syndrome = True and Respiratory Syndrome = False
 8.84% (884/10000) of baseline cases have Viral Syndrome = True and Respiratory Syndrome = False
2. 2001-06-02: SCORE = -3.19604e-08 PVALUE = 0
 1.27971% (7/547) of today’s cases have Age Decile = 10 and Home Latitude = Missing
 0.02% (2/10000) of baseline cases have Age Decile = 10 and Home Latitude = Missing
3. 2001-06-30: SCORE = -2.39821e-07 PVALUE = 0
 1.44% (9/625) of today’s cases have Age Decile = 10
 0.09% (9/10000) of baseline cases have Age Decile = 10
4. 2001-08-08: SCORE = -1.21558e-08 PVALUE = 0
 83.7979% (481/574) of today’s cases have Unknown Syndrome = False
 73.6926% (7370/10001) of baseline cases have Unknown Syndrome = False

WHAT'S STRANGE ABOUT RECENT EVENTS

5. 2001-10-10: SCORE = -1.42315e-06 PVALUE = 0
0.994036% (5/503) of today's cases have Age Decile = 10 and Home Latitude = Missing
0.009998% (1/10002) of baseline cases have Age Decile = 10 and Home Latitude = Missing
6. 2001-12-02: SCORE = -4.31806e-07 PVALUE = 0
14.7059% (70/476) of today's cases have Viral Syndrome = True and Encephalitic Syndrome = False
7.73077% (773/9999) of baseline cases have Viral Syndrome = True and Encephalitic Syndrome = False
7. 2001-12-09: SCORE = -3.31973e-10 PVALUE = 0
8.57788% (38/443) of today's cases have Hospital ID = 1 and Viral Syndrome = True
2.49% (249/10000) of baseline cases have Hospital ID = 1 and Viral Syndrome = True

Rules 2, 3 and 5 are likely due to clerical errors in the data since the rule finds an increase in the number of people between the ages of 100 and 110. Furthermore, the home zip code for these patients appears to be missing in rules 2 and 5. Rule 4 is uninteresting since it indicates that the number of cases without an unknown symptom, which is typically around 73.7%, has experienced a slight increase. For rules 1, 6 and 7 we went back to the original ED data to inspect the text descriptions of the chief complaints for the cases related to these three rules. The symptoms related to Rules 1, 6 and 7 involve dizziness, fever and sore throat. Given that Rules 1, 6 and 7 have dates in winter, along with the symptoms mentioned, we speculate that this anomalous pattern is likely caused by an influenza strain.

We also include results from WSARE 2.0 running on the same data set. Unlike WSARE 3.0, WSARE 2.0 does not have a similar solution to the approach taken in Section 3.2 to deal with new hospitals coming online. However, by using a short enough baseline period, such as the standard baseline of 35, 42, 49, and 56 days prior to the current date, we can capture fairly recent trends and deal with a changing distribution as new hospitals submit data. The results are shown below. Note that we group together identical rules from consecutive days in order to save space.

1. 2001-01-31: SCORE = -8.0763e-07 PVALUE = 0
21.2766% (110/517) of today's cases have Unknown Syndrome = True
12.5884% (267/2121) of baseline cases have Unknown Syndrome = True
2. 2001-05-01: SCORE = -1.0124e-06 PVALUE = 0.001998
18.4739% (92/498) of today's cases have Gender = Male and Home Latitude > 40.5
10.2694% (202/1967) of baseline cases have Gender = Male and Home Latitude > 40.5

Rules 3-6 from 2001-10-28 to 2001-10-31 all have PVALUE = 0 and involve rules with Hospital ID = Missing

7. 2001-11-01: SCORE = -7.78767e-21 PVALUE = 0
5.87084% (30/511) of today's cases have Hospital ID = Missing and Hemorrhagic Syndrome = True
0% (0/1827) of baseline cases have Hospital ID = Missing and Hemorrhagic Syndrome = True

Rules 8-14 from 2001-11-02 to 2001-11-08 all have PVALUE = 0 and have the rule Hospital ID = Missing

Rules 15-37 from 2001-11-09 to 2001-12-02 all have PVALUE = 0 and have the rule Hospital ID = 14

Rules 38-59 from 2001-12-03 to 2001-12-24 all have PVALUE = 0 and have the rule Hospital ID = 50

60. 2001-12-25: SCORE = -2.99132e-09 PVALUE = 0

53.1835% (284/534) of today's cases have Rash Syndrome = False and Unmapped Syndrome = False

39.2165% (911/2323) of baseline cases have Rash Syndrome = False and Unmapped Syndrome = False

Rules 61-63 from 2001-12-26 to 2001-12-30 all have PVALUE = 0 and have the rule Hospital ID = 50

64. 2001-12-31: SCORE = -7.30783e-07 PVALUE = 0

52.071% (352/676) of today's cases have Hemorrhagic Syndrome = True and Unmapped Syndrome = False

41.6113% (1064/2557) of baseline cases have Hemorrhagic Syndrome = True and Unmapped Syndrome = False

From the output above, WSARE 2.0 produces a large number of rules that involves hospital IDs 14 and 50 because those two hospitals start providing data in 2002. These rules typically persist for about a month, at which point the new hospitals begin to appear in the baseline of WSARE 2.0. We speculate that the missing hospital IDs in rules 3-14 are due to hospital 14 coming online and a new hospital code not being available. The other rules produced by WSARE 2.0 are very different from those generated by WSARE 3.0. This difference is likely due to the fact that WSARE 3.0 considers the effects of the environmental attributes. The most interesting rules produced by WSARE 2.0 are rules 2 and 64. Rule 2 highlights the fact that more male patients with a home zip code in the northern half of the city appear in the EDs on 2001-05-01. Rule 64 indicates that an increase in the number of hemorrhagic syndromes have occurred. Both of these rules are unlikely to have been caused by environmental trends; they are simply anomalous patterns when compared against the baseline of WSARE 2.0. From our available resources, we are unable to determine if rules 2 and 64 are truly indicative of an outbreak.

4.3 Results from the Israel Center for Disease Control

The Israel Center for Disease Control evaluated WSARE 3.0 retrospectively using an unusual outbreak of influenza type B that occurred in an elementary school in central Israel (Kaufman et al., 2004). WSARE 3.0 was applied to patient visits to community clinics between the dates of May 24, 2004 to June 11, 2004. The attributes in this data set include the visit date, area code, ICD-9 code, age category, and day of week. The day of week was used as the only environmental attribute. WSARE 3.0 reported two rules with p-values at 0.002 and five other rules with p-values below 0.0001. Two of the five anomalous patterns with p-values below 0.0001 corresponded to the influenza outbreak in the data. The rules that characterized the two anomalous patterns consisted of the same three attributes of ICD-9 code, area code and age category, indicating that an anomalous pattern was found involving children aged 6-14 having viral symptoms within a specific geographic area. WSARE 3.0 detected the outbreak on the second day from its onset. The authors of (Kaufman et al., 2004) found the results from WSARE 3.0 promising and concluded that the algorithm was indeed able to detect an actual outbreak in syndromic surveillance data.

4.4 Summary of Results

Overall, WSARE 2.0 and 3.0 have been demonstrated to be more effective than univariate methods at finding anomalous patterns in multivariate, categorical data. The advantage that the WSARE algorithms have over univariate methods is their ability to identify the combination of attributes that characterize the most anomalous groups in the data rather than relying on a user to specify

beforehand which combination of characteristics to monitor. WSARE 3.0 has a further advantage in its ability to account for temporal trends when producing the baseline distribution while WSARE 2.0 can be thrown off by these temporal trends when it uses raw historical data for the baseline.

We would like to emphasize the fact that WSARE 3.0 is not necessarily the best version of WSARE in all cases. WSARE 3.0 needs a large amount of data in order to learn the structure and parameters of its Bayesian network reliably, particularly if there are many attributes in the data. If WSARE 3.0 is intended to model long term trends such as seasonal fluctuations, several years worth of historical data are needed. Large amounts of historical data are not available in many cases, such as when a syndromic surveillance system needs to be set up from scratch in a few months for a major event like the Olympic games. In these scenarios, WSARE 2.0 may have an advantage over WSARE 3.0. This disadvantage of WSARE 3.0 highlights the fact that the learned Bayesian network only stores the posterior mean in the conditional probability tables of each node. Future work on WSARE 3.0 will involve accounting for the variance of the network parameters in the p-value calculation, perhaps using the approaches proposed by van Allen (2000), van Allen et al. (2001), and Singh (2004).

Moreover, WSARE 3.0 assumes that the environmental attributes are the only source of variation in the baseline distribution. If other hidden variables cause a significant amount of noise in the baseline, then WSARE 3.0 will not be very effective. In this situation, a better approach might be to use WSARE 2.0 with a baseline of raw historical data from a very recent time period. Finally, we do not recommend using WSARE 2.5 because the algorithm is unable to make predictions for days in which the combination of environmental attributes do not exist in historical data. The Bayesian network used by WSARE 3.0 is able to handle such situations and WSARE 3.0 effectively supersedes WSARE 2.5.

5. Finding Anomalous Patterns in Real-Valued Data

The WSARE algorithm can only be used on categorical data sets. If the data is entirely real-valued, the attributes can certainly be discretized in a pre-processing step before WSARE operates on the data. Discretization, however, treats all data points in the same discretization bin identically; the distances between data points in the same bin are lost. If these distances are important, then a real-valued version of WSARE is needed. Fortunately, the spatial scan statistic (Kulldorff, 1997) can be considered as the real-valued analog of WSARE.

The spatial scan statistic works on a geographic area A in which there is an underlying population n and within this population there is a count c of interest. The distribution of the counts c is assumed to follow either a Bernoulli model or a Poisson model. A window of variable size and shape then passes through the geographic area A . The crucial characteristic of this window is that the union of the areas covered by the window is the entire area A . Existing spatial scan statistic applications typically use window shapes of circles (Kulldorff, 1999) although ellipses (Kulldorff et al., 2002) and rectangles (Neill and Moore, 2004) have also been used. In order to set up the scan statistic, we need to define p as the probability of being a “count” within the scanning window. Furthermore, let q be the probability of being a “count” outside of the scanning window. Under the null hypothesis, $p = q$ while the alternative hypothesis is $p > q$. The spatial scan statistic then consists of the maximum likelihood ratio between L_W , the likelihood of the counts in the scanning window area W , and L_0 , the likelihood under the null hypothesis. Equation 1 illustrates the spatial scan statistic in its general form, using the term W for the zone covered by a scanning window and

\mathcal{W} for the entire collection of zones:

$$S_{\mathcal{W}} = \max_{W \in \mathcal{W}} \frac{L(W)}{L_0}. \quad (1)$$

Since an analytical form for the distribution of the spatial scan statistic is not available, a Monte Carlo simulation is needed to obtain the significance of the hypothesis test. Typically 999 or 9999 replications of the data set are used for the simulation. In terms of computational complexity, the bottleneck for the algorithm is the Monte Carlo simulation.

The spatial scan statistic has been extended to three dimensions in the space-time scan statistic (Kulldorff, 1999, 2001). Instead of using a circular window over space, the scanning window is now a cylinder, with its circular base for the spatial dimension and its height over a time interval. Cylinders of varying heights and base radii are moved through space and time to find potential disease clusters.

Naive implementations of the spatial scan statistic and the space-time scan statistic are too computationally expensive for large data sets. Assuming that the circular windows are centered on an $N \times N$ grid and the dimensionality is D , the complexity is $O(RN^{2D})$ where R is the number of Monte Carlo simulations. Neill et al. (2005) have developed a fast spatial scan using overlap-kd trees that can reduce the complexity to $O(R(N \log N)^D)$ in the best case. The algorithms discussed so far find abnormally high density regions in data sets that are entirely real-valued. Efficiently finding anomalous patterns in a data set with a mixture of categorical and real-valued attributes remains an open problem.

6. Related Work

The task of detecting anomalous events in data is most commonly associated with monitoring systems. As a result, related work can be found in the domains of computer security, fraud detection, Topic Detection and Tracking (TDT) and fMRI analysis. In computer security, anomaly detection has been most prominent in intrusion detection systems, which identify intrusions by distinguishing between normal system behavior and behavior when security has been compromised (Lane and Brodley, 1999; Warrender et al., 1999; Eskin, 2000; Lee et al., 2000; Maxion and Tan, 2002; Kruegel and Vigna, 2003). In other related security work, Cabuk et al. (2004) describe methods to detect IP covert timing channels, which surreptitiously use the arrival pattern of packets to send information. As in computer security, automated fraud detection systems differentiate between normal and unusual activity on a variety of data such as cellular phone calls (Fawcett and Provost, 1997) and automobile insurance fraud (Phua et al., 2004). TDT is the task of identifying the earliest report of a previously unseen news story from a sequence of news stories. Clustering approaches are typically used in TDT (Yang et al., 1998; Zhang et al., 2005). Finally, anomalous event detection has also been used in fMRI analysis to identify regions of increased brain activity corresponding to given cognitive tasks (Neill et al., 2005).

In general, WSARE can be applied to data from these different domains as long as the data and the anomalous events satisfy several criteria. WSARE is intended to operate on categorical, case-level records in which the presence of a record can be considered an event. For instance, in ED data, an event is defined as the appearance of a person at the ED since it provides a signal of the community health and we are interested in the characteristics of that person. Secondly, WSARE only finds differences between the recent data and the baseline data. If we consider the baseline

data to be a “class”, then WSARE looks for deviations from a single class. Some domains, such as TDT, require comparisons between several classes. For instance, the current news story needs to be compared against several categories of news stories. Thirdly, as was discussed in Section 2.6, WSARE’s running time depends on the number of attributes and the number of values each attribute can take. If the number of attributes and the number of values for each attribute are too high, WSARE may not finish in a reasonable amount of time. Some domains require the running time of the detection algorithm to be a few seconds or less in order for the entire detection system to be effective. In these situations, using WSARE is not appropriate. On the other hand, for domains such as biosurveillance, the running time of WSARE is acceptable since it takes approximately a minute to a few minutes to complete on real ED data sets. Finally, WSARE treats each record in the data independently of the other records. If a sequence of records is highly indicative of, for instance, a security breach in a network, WSARE will not be able to detect this pattern.

Other related work can also be found in the area of stream mining. In stream mining, the focus is on the online processing of large amounts of data as it arrives. Many algorithms have been developed to detect anomalies in the current stream of data. Ma and Perkins (2003) develop a novelty detection algorithm based on online support vector regression. Anomalies can also be characterized by an abnormal burst of data. The technique described by Zhu and Shasha (2003) simultaneously monitors windows of different sizes and reports those that have an abnormal aggregation of data. A density estimation approach is used by Aggarwal (2003) to help visualize both spatial and temporal trends in evolving data streams. Finally, Hulten et al. (2001) present an efficient algorithm for mining decision trees from continuously changing data streams. While this work is primarily concerned with maintaining an up-to-date concept, detecting concept drift is similar to detecting changes in a data stream. WSARE 3.0 cannot be directly applied to stream mining because the amount of historical data needed to create the baseline distribution is typically not accessible in a stream mining context. However, WSARE 2.0 could possibly be modified for a stream mining application.

In the following paragraphs, we will briefly review methods that have been used for the detection of disease outbreaks. Readers interested in a detailed survey of biosurveillance methods can be found in (Wong, 2004) and (Moore et al., 2003). The majority of detection algorithms in biosurveillance operate on univariate time series data. Many of these univariate algorithms have been taken from the field of Statistical Quality Control and directly applied to biosurveillance. The three most common techniques from Statistical Quality Control include the Shewhart control chart (Montgomery, 2001), CUSUM (Page, 1954; Hutwagner et al., 2003), and EWMA (Roberts, 1959; Williamson and Hudson, 1999). Although these three algorithms are simple to implement, they have difficulty dealing with temporal trends. Univariate algorithms based on regression and time series models, on the other hand, are able to model explicitly the seasonal and day of week effects in the data. The Serfling method (Serfling, 1963) uses sinusoidal components in its regression equation to model the seasonal fluctuations for influenza. A Poisson regression model that included a day of week term as a covariate was demonstrated to be a fairly capable detector in (Buckeridge et al., 2005). As for time series models, the ARIMA and SARIMA models (Choi and Thacker, 1981; Watier et al., 1991; Reis and Mandl, 2003) are commonly used in biosurveillance to deal with temporal trends. Recently, wavelets (Goldenberg et al., 2002; Zhang et al., 2003) have been used as a preprocessing step to handle temporal fluctuations including unusually low values due to holidays.

The most common algorithm used in biosurveillance of spatial data is the Spatial Scan Statistic (Kulldorff, 1997), which has already been discussed. The Spatial Scan Statistic has been generalized to include a time dimension (Kulldorff, 2001) such that the algorithm searches for cylinders in

spatio-temporal data. Recent work has improved the speed of the Spatial Scan method using an overlap-kd tree structure (Neill and Moore, 2004; Neill et al., 2005).

The algorithms mentioned thus far have only looked at either univariate or spatial data. Only a few multivariate biosurveillance algorithms that consider spatial, temporal, demographic, and symptomatic attributes for individual patient cases currently exist. BCD (Buckeridge et al., 2005) is a multivariate changepoint detection algorithm that monitors in a frequentist manner whether a Bayesian network learned from past data (during a “safe” training period) appears to have a distribution that differs from the distribution of more recent data. If so, then an anomaly may have occurred. The Bayesian Aerosol Release Detector (BARD) (Hogan et al., 2004) is an algorithm specifically designed to detect an outbreak of inhalational anthrax due to atmospheric dispersion of anthrax spores. BARD combines information from ED visits, recent meteorological data, and spatial and population information about the region being monitored in order to determine if an anthrax attack has occurred. Finally, PANDA (Cooper et al., 2004) is a population-based anomaly detection algorithm that uses a massive causal Bayesian network to model each individual in the region under surveillance. By modeling at the individual level, PANDA is able to coherently represent different types of background knowledge in its model. For example, spatio-temporal assumptions about a disease outbreak can be incorporated as prior knowledge. In addition, the characteristics of each individual, such as their age, gender, home zip, symptom information and admission date to the ED can be used to derive a posterior probability of an outbreak.

There are two algorithms that are similar to the approach taken by WSARE. Contrast set mining (Bay and Pazzani, 1999) finds rules that distinguish between two or more groups using a pruning algorithm to reduce the exponential search space. This optimization prunes away rules whose counts are too small to yield a valid Chi Square test. Many of these rules are interesting to WSARE. Multiple hypothesis testing problems are addressed in contrast set mining through a Bonferroni correction. In health care, Brossette et al. use association rules for hospital infection control and public health surveillance (Brossette et al., 1998). Their work is similar to WSARE 2.0 (Wong et al., 2002), with the main difference being the additional steps of the randomization test and FDR in WSARE.

7. Conclusions

WSARE approaches the problem of early outbreak detection on multivariate surveillance data using two key components. The first component is association rule search, which is used to find anomalous patterns between a recent data set and a baseline data set. The contribution of this rule search is best seen by considering the alternate approach of monitoring a univariate signal. If an attribute or combination of attributes is known to be an effective signal for the presence of a certain disease, then a univariate detector or a suite of univariate detectors that monitors this signal will be an effective early warning detector for that specific disease. However, if such a signal is not known beforehand, then the association rule search will determine which attributes are of interest. We intend WSARE to be a general purpose safety net to be used in combination with a suite of specific disease detectors. Thus, the key to this safety net is to perform non-specific disease detection and notice any unexpected patterns.

With this perspective in mind, the fundamental assumption to our association rule approach is that an outbreak in its early stages will manifest itself in categorical surveillance data as an anomalous cluster in attribute space. For instance, a localized gastrointestinal outbreak originating at a

popular restaurant in zipcode X would likely cause an upswing in diarrhea cases involving people with home zipcode X. These cases would appear as a cluster in the categorical attributes of *Home Zip Code = X* and *Symptom = Diarrhea*. The rule search allows us to find the combination of attributes that characterize the set of cases from recent data that are most anomalous when compared to the baseline data. The nature of the rule search, however, introduces the problem of multiple hypothesis testing to the algorithm. Even with purely random data, the best scoring rule may seem like a truly significant anomalous pattern. We are careful to evaluate the statistical significance of the best scoring rule using a randomization test in which the null hypothesis is the independence of date and case attributes.

The second major component of WSARE is the use of a Bayesian network to model a baseline that changes due to temporal fluctuations such as seasonal trends and weekend versus weekday effects. In WSARE 3.0, attributes are divided into environmental and response attributes. Environmental attributes, such as season and day of week, are attributes which are responsible for the temporal trends while response attributes are the non-environmental attributes. When the Bayesian network structure is learned, the environmental attributes are not permitted to have parents because we are not interested in predicting their distributions. Instead, we want to determine how the environmental attributes affect the distributions of the response attributes. WSARE 3.0 operates on an assumption that the environmental attributes account for the majority of the variation in the data. Under this assumption, the ratios compared in the rule search should remain reasonably stable over historical time periods with similar environmental attribute values. As an example, if the current day is a winter Friday and we use season and day of week as environmental attributes, then the fraction of male senior citizens, for instance, showing up at an ED to the total number of patients should remain roughly stable over all winter Fridays in the historical period over which the Bayesian network is learned. Once the Bayesian network structure is learned, it represents the joint probability distribution of the baseline. We can then condition on the environmental attributes to produce the baseline given the environment for the current day.

Multivariate surveillance data with known outbreak periods is extremely difficult to obtain. As a result, we resorted to evaluating WSARE on simulated data. Although the simulators do not reflect real life, detecting an outbreak in our simulated data sets is a challenging problem for any detection algorithm. We evaluated WSARE on the CityBN simulator, which was implemented to generate surveillance data which contained temporal fluctuations due to day of week effects and seasonal variations of background illnesses such as flu, food poisoning and allergies. Despite the fact that the environmental attributes used by WSARE 3.0 did not account for all of the variation in the data, WSARE 3.0 detected the anthrax outbreaks with nearly the optimal detection time and a very low false positive rate. We show that WSARE 3.0 outperformed three common univariate detection algorithms in terms of false positives per month and detection time. WSARE 3.0 also produced a better AMOC curve than WSARE 2.0 because the latter was thrown off by the temporal trends in the data. Finally, the Bayesian network provided some smoothing to the baseline distribution which enhanced WSARE 3.0's detection capability as compared to that of WSARE 2.5.

WSARE has been demonstrated to outperform traditional univariate methods on simulated data in terms of false positives per month and detection time. Its performance on real world data requires further evaluation. Currently, WSARE is part of the collection of biosurveillance algorithms in the RODS system (Real-time Outbreak Detection System, 2004). WSARE 2.0 was deployed to monitor ED cases in western Pennsylvania and Utah. It was also used during the 2002 Salt Lake City winter

Olympics. WSARE 3.0 is currently being used as a tool for analysis of public health data by several American state health departments and by the Israel Center for Disease Control.

Acknowledgments

This research was supported by DARPA grant F30602-01-2-0550, the State of Pennsylvania, and by the National Science Foundation grant IIS-0325581. Many thanks to Rich Tsui, Bob Olszewski, Jeremy Espino, Jeff Schneider and Robin Sabhnani for their helpful comments and technical expertise.

References

- 45 CFR Parts 160 through 164, April 2003.
Available at <http://www.hhs.gov/ocr/combinedregtext.pdf>.
- Charu C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 575–586, New York, NY, 2003. ACM Press.
- Stephen D. Bay and Michael J. Pazzani. Detecting change in categorical data: Mining contrast sets. In *Knowledge Discovery and Data Mining*, pages 302–306, 1999.
- Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B.*, 57:289–300, 1995.
- Chris M. Bishop. Novelty detection and neural network validation. *IEEE Proceedings - Vision, Image and Signal Processing*, 141(4):217–222, August 1994.
- Carlo E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.
- George E. P. Box and Gwilym M. Jenkins. *Time series analysis: Forecasting and control*. Holden-Day, San Francisco, 1976.
- Stephen E. Brossette, Alan P. Sprague, J. Michael Hardin, Ken B. Waites, Warren T. Jones, and Stephen A. Moser. Association rules and data mining in hospital infection control and public health surveillance. *Journal of the American Medical Informatics Association*, 5:373–381, 1998.
- David L. Buckeridge, Howard Burkom, Murray Campbell, William R. Hogan, and Andrew W. Moore. Algorithms for rapid outbreak detection: a research synthesis. *Biomedical Informatics*, 38(2):99–113, 2005.
- Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP covert timing channels: design and detection. In *Proceedings of the 11th ACM conference on Computer and Communications Security*, pages 178–187, New York, NY, 2004. ACM Press.

- Keewhan Choi and Stephen B. Thacker. An evaluation of influenza mortality surveillance, 1962-1979 I. time series forecasts of expected pneumonia and influenza deaths. *American Journal of Epidemiology*, 113(3):215–226, 1981.
- Gregory F. Cooper, Denver H. Dash, John D. Levander, Weng-Keen Wong, William R. Hogan, and Michael M. Wagner. Bayesian biosurveillance of disease outbreaks. In Max Chickering and Joseph Halpern, editors, *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, pages 94–103, Banff, Alberta, Canada, 2004. AUAI Press.
- Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the 2000 International Conference on Machine Learning (ICML-2000)*, Palo Alto, CA, July 2000.
- Tom Fawcett and Foster Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.
- Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. In Chaudhuri and Madigan, editors, *Proceedings on the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, San Diego, CA, 1999. URL citeseer.nj.nec.com/fawcett99activity.html.
- Anna Goldenberg, Galit Shmueli, and Rich Caruana. Using grocery sales data for the detection of bio-terrorist attacks. Submitted to *Statistics in Medicine*, 2003.
- Anna Goldenberg, Galit Shmueli, Richard A. Caruana, and Stephen E. Fienberg. Early statistical detection of anthrax outbreaks by tracking over-the-counter medication sales. *Proceedings of the National Academy of Sciences*, 99(8):5237–5240, April 2002. <http://www.pnas.org/cgi/doi/10.1073/pnas.042117499>.
- Phillip Good. *Permutation Tests - A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer-Verlag, New York, 2nd edition, 2000.
- Greg Hamerly and Charles Elkan. Bayesian approaches to failure prediction for disk drives. In *Proceedings of the eighteenth international conference on machine learning*, pages 202–209. Morgan Kaufmann, San Francisco, CA, 2001.
- William R. Hogan, Gregory F. Cooper, Garrick L. Wallstrom, and Michael M. Wagner. The Bayesian aerosol release detector. In *Proceedings of the Third National Syndromic Surveillance Conference [CD-ROM]*, Boston, MA, 2004. Fleetwood Multimedia, Inc.
- Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA, 2001. ACM Press. URL citeseer.ist.psu.edu/hulten01mining.html.
- Lori Hutwagner, William Thompson, G. Matthew Seeman, and Tracee Treadwell. The bioterrorism preparedness and response early aberration reporting system (EARS). *Journal of Urban Health*, 80(2):i89–i96, 2003.

- Zalman Kaufman, Erica Cohen, Tamar Peled-Leviatan, Hanna Lavi, Gali Aharonowitz, Rita Dichtiar, Michal Bromberg, Ofra Havkin, Yael Shalev, Rachel Marom, Varda Shalev, Joshua Shemer, and Manfred S Green. Evaluation of syndromic surveillance for early detection of bioterrorism using a localized, summer outbreak of Influenza B. In *Proceedings of the Third National Syndromic Surveillance Conference [CD-ROM]*, Boston, MA, 2004. Fleetwood Multimedia Inc. Poster Presentation.
- Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS '03)*, pages 251–261, Washington, DC, October 2003. ACM Press.
- Martin Kulldorff. A spatial scan statistic. *Communications in Statistics: Theory and Methods*, 26(6):1481–1496, 1997.
- Martin Kulldorff. Spatial scan statistics: models, calculations, and applications. In J. Glaz and N. Balakrishnan, editors, *Scan Statistics and Applications*, pages 303–322. Birkhauser, Boston, MA, 1999.
- Martin Kulldorff. Prospective time periodic geographical disease surveillance using a scan statistic. *Journal of the Royal Statistical Society, Series A*, 164:61–72, 2001.
- Martin Kulldorff, Lan Huang, and Linda Pickle. An elliptic spatial scan statistic and its application to breast cancer mortality data in the northeastern united states. In *Proceedings of the National Syndromic Surveillance Conference*, 2002.
- Terran Lane and Carla E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2:295–331, 1999.
- Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review*, 14(6):533–567, 2000. URL citeseer.ist.psu.edu/lee00adaptive.html.
- Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *Proceedings on the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 613–618, New York, NY, 2003. ACM Press.
- Oded Maron and Andrew W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1-5):193–225, 1997.
- Roy A. Maxion and Kymie M.C. Tan. Anomaly detection in embedded systems. *IEEE Trans. Comput.*, 51(2):108–120, 2002. ISSN 0018-9340.
- Christopher J. Miller, Christopher Genovese, Robert C. Nichol, Larry Wasserman, Andrew Connolly, Daniel Reichart, Andrew Hopkins, Jeff Schneider, and Andrew Moore. Controlling the false-discovery rate in astrophysical data analysis. *The Astronomical Journal*, 122:3492–3505, Dec 2001.
- Douglas C. Montgomery. *Introduction to Statistical Quality Control*. John Wiley and Sons, Inc., 4th edition, 2001.

- Andrew Moore and Mary Soon Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, March 1998.
- Andrew Moore and Weng-Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning*, pages 552–559, Menlo Park, CA, August 2003. AAAI Press.
- Andrew W. Moore, Gregory F. Cooper, Rich Tsui, and Michael M. Wagner. Summary of biosurveillance-related technologies. Technical report, Realtime Outbreak and Disease Surveillance Laboratory, University of Pittsburgh, 2003. Available at <http://www.autonlab.org/autonweb/showPaper.jsp?ID=moore-biosurv>.
- Farzad Mostashari and Jessica Hartman. Syndromic surveillance: a local perspective. *Journal of Urban Health*, 80(2):i1–i7, 2003.
- Daniel B. Neill and Andrew W. Moore. A fast multi-resolution method for detection of significant spatial disease clusters. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
- Daniel B. Neill, Andrew W. Moore, Francisco Pereira, and Tom Mitchell. Detecting significant multidimensional spatial clusters. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.
- E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1):100–115, 1954.
- Clifton Phua, Daminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *ACM SIGKDD Explorations Newsletter Special issue on learning from imbalanced datasets*, 6(1):50–59, 2004.
- Real-time Outbreak Detection System, 2004. Online at <http://www.health.pitt.edu/rods/default.htm>.
- Ben Y. Reis and Kenneth D. Mandl. Time series modeling for syndromic surveillance. *BMC Medical Informatics and Decision Making*, 3(2), 2003. <http://www.biomedcentral.com/1472-6947/3/2>.
- S. W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1:239–250, 1959.
- Robert E. Serfling. Methods for current statistical analysis of excess pneumonia-influenza deaths. *Public Health Reports*, 78:494–506, 1963.
- Ajit P. Singh. What to do when you don't have much data: Issues in small sample parameter learning in Bayesian Networks. Master's thesis, Dept. of Computing Science, University of Alberta, 2004.
- Daniel M. Sosin. Draft framework for evaluating syndromic surveillance systems. *Journal of Urban Health*, 80(2):i8–i13, 2003.
- Fu-Chiang Tsui, Michael M. Wagner, Virginia Dato, and Chung-Chou Ho Chang. Value of ICD-9-coded chief complaints for detection of epidemics. In S Bakken, editor, *Journal of the American*

- Medical Informatics Association, Supplement i ssue on the Proceedings of the Annual Fall Symposium of the American Medical Inf ormatics Association*, pages 711–715. Hanley and Belfus, Inc, 2001.
- Tim van Allen. Handling uncertainty when you’re handling uncertainty: Model selection and error bars for belief networks. Master’s thesis, Dept. of Computing Science, University of Alberta, 2000.
- Tim van Allen, Russell Greiner, and Peter Hooper. Bayesian error-bars for belief net inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, Aug 2001.
- Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 133–145. IEEE Computer Society, 1999.
- Laurence Watier, Sylvia Richardson, and Bruno Hubert. A time series construction of an alert threshold with application to s. bovismorbificans in france. *Statistics in Medicine*, 10:1493–1509, 1991.
- G. David Williamson and Ginner Weatherby Hudson. A monitoring system for detecting aberrations in public health surveillance reports. *Statistics in Medicine*, 18:3283–3298, 1999.
- Weng-Keen Wong. *Data mining for early disease outbreak detection*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2004.
- Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In Tom Fawcett and Nina Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 808–815, Menlo Park, California, August 2003. AAAI Press.
- Weng-Keen Wong, Andrew W. Moore, Gregory Cooper, and Michael Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 217–223. MIT Press, 2002.
- Yiming Yang, Tom Pierce, and Jiame Carbonell. A study on retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 28–36, New York, NY, 1998. ACM Press.
- Jian Zhang, Zoubin Ghahramani, and Yiming Yang. A probabilistic model for online document clustering with application to novelty detection. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.
- Jun Zhang, Fu-Chiang Tsui, Michael M. Wagner, and William R. Hogan. Detection of outbreaks from time series data using wavelet transform. In *Proc AMIA Fall Symp*, pages 748–752. Omni Press CS, 2003.
- Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 336–345, New York, NY, 2003. ACM Press.

Change Point Problems in Linear Dynamical Systems

Onno Zoeter

SNN, Biophysics

Radboud University Nijmegen

Geert Grooteplein 21

NL 6525 EZ, Nijmegen, The Netherlands

O.ZOETER@SCIENCE.RU.NL

Tom Heskes

Computer Science

Radboud University Nijmegen

Toernooiveld 1

NL 6525 ED Nijmegen, The Netherlands

T.HESKES@SCIENCE.RU.NL

Editor: Donald Geman

Abstract

We study the problem of learning two regimes (we have a normal and a pre-fault regime in mind) based on a train set of non-Markovian observation sequences. Key to the model is that we assume that once the system switches from the normal to the pre-fault regime it cannot restore and will eventually result in a fault. We refer to the particular setting as *semi-supervised* since we assume the only information given to the learner is whether a particular sequence ended with a stop (implying that the sequence was generated by the normal regime) or with a fault (implying that there was a switch from the normal to the fault regime). In the latter case the particular time point at which a switch occurred is not known.

The underlying model used is a *switching linear dynamical system (SLDS)*. The constraints in the regime transition probabilities result in an exact inference procedure that scales quadratically with the length of a sequence. Maximum a posteriori (MAP) parameter estimates can be found using an expectation maximization (EM) algorithm with this inference algorithm in the E-step. For long sequences this will not be practically feasible and an approximate inference and an approximate EM procedure is called for. We describe a flexible class of approximations corresponding to different choices of clusters in a Kikuchi free energy with weak consistency constraints.

Keywords: change point problems, switching linear dynamical systems, strong junction trees, approximate inference, expectation propagation, Kikuchi free energies

1. Introduction

In this article we investigate the problem of detecting a change in a dynamical system. An obvious practical application of such a model is the prediction of oncoming faults in an industrial process.

For simplicity the problem and algorithms are outlined for a model with four regimes, *normal*, *pre-fault*, *stop*, and *fault*, in Section 2 the extension to more regimes is discussed. The stop and fault regimes are special in the sense that they are *absorbing*. If the system reaches one of these states the process stops. A key assumption in the problem is that once the system reaches a pre-fault state, it can never recover.

The setup could be considered as a *change point problem*, although the name change point problem usually refers to a problem where the observations are independent if the underlying model parameters are known. In such settings the challenge is to determine if and where the parameters change their value. See Krishnaiah and Miao (1988) for a description of change point problems and references.

In this article we will be interested in slightly more complex problems where the observations are dependent, even if the parameters are known. The observations in the time-series are not assumed to be Markov. Instead, they are noisy observations of a latent first order Markov process.

The model discussed in this paper can be identified as a *switching linear dynamical system* (SLDS), with restricted dynamics in the regime indicators. The SLDS is a discrete time model and consists of T , d dimensional observations $\mathbf{y}_{1:T}$ and T , q dimensional latent states $\mathbf{x}_{1:T}$. The regime in every time-step is determined by (typically unobserved) discrete switches $s_{1:T}$. For $1 < t \leq T$, s_t is either normal or pre-fault. The last discrete indicator s_{T+1} is either a stop or a fault.

Within every regime the state transition and the observation model are linear Gaussian, and may differ per regime:

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{x}_{t-1}, s_t, \theta) &= \mathcal{N}(\mathbf{x}_t; A_{s_t} \mathbf{x}_{t-1}, Q_{s_t}), \\ p(\mathbf{y}_t | \mathbf{x}_t, s_t, \theta) &= \mathcal{N}(\mathbf{y}_t; C_{s_t} \mathbf{x}_t + \mu_{s_t}, R_{s_t}). \end{aligned}$$

In the above $\mathcal{N}(\cdot; \cdot, \cdot)$ denotes the Gaussian density function

$$\mathcal{N}(\mathbf{x}; \mathbf{m}, V) \equiv (2\pi)^{-(d/2)} |V|^{-1/2} \exp \left[-\frac{1}{2} (\mathbf{x} - \mathbf{m})^\top V^{-1} (\mathbf{x} - \mathbf{m}) \right].$$

The determinant of matrix V is denoted as $|V|$. The set of parameters in the model is denoted by θ . As mentioned the current regime is encoded by discrete random variables $s_{1:T}$ and are assumed to follow a first order transition model

$$p(s_t | s_{t-1}, \theta) = \Pi_{s_{t-1} \rightarrow s_t}.$$

The special characteristics of the regimes and their transitions are reflected by zeros in $\Pi_{s_{t-1} \rightarrow s_t}$, i.e. denoting the possible states (normal, fault, etc.) by their initial letter, we require $\Pi_{n \rightarrow f} = 0$, $\Pi_{p \rightarrow n} = 0$, $\Pi_{p \rightarrow s} = 0$, $\Pi_{s \rightarrow j} = 0$, for all $j \neq s$ and $\Pi_{f \rightarrow j} = 0$, for all $j \neq f$.

The first regime is always normal, i.e. $s_1 = n$, and the first latent state is drawn from a Gaussian prior

$$p(\mathbf{x}_1 | s_1 = n, \theta) = \mathcal{N}(\mathbf{x}_1; \mathbf{m}_1, V_1).$$

With these choices the entire model is *conditional Gaussian*; conditioned on the discrete variables $s_{1:T}$, the remaining variables are jointly Gaussian distributed. The conditional independencies implied by the model are depicted as a dynamical Bayesian network (Pearl, 1988) in Figure 1.

One of the properties of the conditional Gaussian distribution which leads often to computational problems is that it is not closed under marginalization. For instance, the state posterior over \mathbf{x}_t given all observations is

$$\sum_{s_{1:T}} \int p(s_{1:T}, \mathbf{x}_{1:T} | \mathbf{y}_{1:T}, \theta) d\mathbf{x}_{1:t-1, t+1:T} = \sum_{s_{1:T}} p(\mathbf{x}_t | s_{1:T}, \mathbf{y}_{1:T}, \theta) p(s_{1:T} | \mathbf{y}_{1:T}, \theta),$$

which is not a conditional Gaussian, but a mixture of Gaussians with M^T components, with M the number of possible regimes in the system. However, as we will discuss in the next section

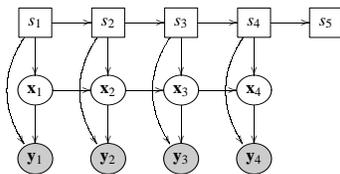


Figure 1: The dynamic Bayesian network for a switching linear dynamical system with four observations. Square nodes denote discrete, and ovals denote continuous random variables. Shading emphasizes that a particular variable is observed.

the assumption that a system cannot restore from a pre-fault state to a normal state results in a considerable simplification, since many of these components have zero weight.

In Section 3 we review how this sparsity can be exploited in an inference algorithm. As the basis for an EM algorithm it can then straightforwardly be used to compute MAP estimates for the model parameters. The exact inference algorithm has running time $O(T^2)$. Hence for relatively short sequences the constrained transition model makes exact inference feasible. However for larger sequences the exact inference algorithm will be inappropriate. In Section 5 we introduce a flexible class of approximations which can be interpreted as a generalization of *expectation propagation* (Minka, 2001). It has running time $O(T\kappa)$, with $0 \leq \kappa \leq \lceil \frac{T-2}{2} \rceil$, an integer parameter that can be set according to the available computational resources. With $\kappa = 0$ the approximation is equivalent to an iterated version (Heskes and Zoeter, 2002; Zoeter and Heskes, 2005) of *generalized pseudo Bayes 2* (Bar-Shalom and Li, 1993) with $\kappa = \lceil \frac{T-2}{2} \rceil$ exact inference is recovered. Section 6 discusses experiments with inference and MAP parameter estimation on synthetic data and a change point problem in EMG analysis.

2. Benefits of the Constrained Regime Transition Probabilities

An interesting aspect of the model introduced in Section 1 is that, by the restriction in the regime transitions, the number of possible regimes histories is considerably less than the 2^T possible histories which would be implied by a system with unconstrained transitions (see e.g. Cemgil et al., 2004; Fearnhead, 2003). If the absorbing state s_{T+1} is not observed, there are T possible regime histories in the current model. One normal sequence $s_{1:T} = n$, and $T - 1$ fault sequences: $s_{1:\tau} = n, s_{\tau+1:T} = p$, with $1 \leq \tau \leq T - 1$. In the remainder of this paper we let τ denote the time-slice up to and including which the regime has been normal, i.e. with $\tau = T$ the entire sequence was normal. Under our assumptions, a fault has to be preceded by at least one pre-fault state, so if s_{T+1} is observed to be a fault the entirely normal sequence gets zero weight. So with s_{T+1} observed to be a fault the number of the possible histories is $T - 1$. If s_{T+1} is observed to be a stop, then only the normal sequence has nonzero probability.

If the parameters in the model, θ , are known, the exact posterior over a continuous state

$$p(\mathbf{x}_t | \mathbf{y}_{1:T}, s_{T+1} = f, \theta)$$

is a mixture of Gaussians with $T - 1$ components, one for every regime history, and can be obtained by running the traditional Kalman filter and smoother $T - 1$ times. In fact the posteriors can be

computed in a slightly faster way by computing shared partial results only once. This algorithm is introduced in Section 3, and will form a suitable basis for the approximate algorithm from Section 5. Although we do not expect exact inference to be practical for large T , we can compare approximations with exact results for larger examples than in the regular SLDS case.

The restriction that there are only two non-absorbing regimes is only made for clarity of the exposition. In general the model has M non-absorbing regimes that form stages. No stage can be skipped, and once the system has advanced to the new stage it cannot recover to a previous one. The number of regime histories with non-zero probability in such a system is less than or equal to T^{M-1} . This can be seen by a simple inductive argument: if $M = 1$ there is only one possible history. If $M > 1$ there are $T - (M - 1) + 1 < T$ possible starting points for the M -th regime (including the starting point $T + 1$, i.e. when regime M does not occur). The $M - 1$ steps are deducted since the system needs at least $M - 1$ steps to reach the M -th regime. Once the start of the M -th regime is fixed, we have a smaller problem with $M - 1$ regimes of length at most T . So the number of distinct regime histories is bounded by $T \times T^{M-2}$. In principle this is still polynomial in T and for small M and limited T exact posteriors could be computed, but obviously the need for approximations is stronger with complex models.

3. Inference

In this section we will introduce the exact recursive inference algorithm as a special case of the *sum-product algorithm* (Kschischang et al., 2001). At this point we assume θ known, leaving the MAP estimation problem to Section 4.

We are interested in one-slice and two-slice posteriors, $p(s_t, \mathbf{x}_t | \mathbf{y}_{1:T}, \theta)$ and $p(s_{t-1,t}, \mathbf{x}_{t-1,t} | \mathbf{y}_{1:T}, \theta)$ respectively.

By defining $\mathbf{u}_t \equiv \{s_t, \mathbf{x}_t\}$ we obtain a model that has the same conditional independence structure as the linear dynamical system and the HMM. From time to time we will use a sum notation to denote both the summation over the domain of the discrete variables, and the integration over the domain of the continuous variables in \mathbf{u}_t . The computational complexity in the current case is due to the parametric form of the (conditional) distributions over \mathbf{u}_t as discussed in Section 1.

Assuming θ and $\mathbf{y}_{1:T}$ fixed and given, the joint probability distribution over all the variables in the model can be written as a product of *factors*

$$p(s_{1:T+1}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T} | \theta) = \prod_{t=1}^{T+1} \psi_t(\mathbf{u}_{t-1,t}),$$

with

$$\begin{aligned} \psi_1(\mathbf{u}_1) &\equiv p(s_1 | \theta) p(\mathbf{x}_1 | s_1, \theta) p(\mathbf{y}_1 | \mathbf{x}_1, s_1, \theta), \\ \psi_t(\mathbf{u}_{t-1,t}) &\equiv p(s_t | s_{t-1}, \theta) p(\mathbf{x}_t | \mathbf{x}_{t-1}, s_t, \theta) p(\mathbf{y}_t | \mathbf{x}_t, s_t, \theta) \quad \text{for } t = 2, \dots, T, \\ \psi_{T+1}(s_{T,T+1}) &\equiv p(s_{T+1} | s_T, \theta), \end{aligned} \tag{1}$$

and $\mathbf{u}_0 \equiv \emptyset$ and $\mathbf{u}_{T+1} \equiv s_{T+1}$. The *factor graph* (Kschischang et al., 2001) implied by this choice of factors is shown in Figure 2. Note that we have simplified the figure by not showing the observations $\mathbf{y}_{1:T}$. These are always observed and are incorporated in the factors.

The sum-product algorithm implied by the factor graph from Figure 2 is presented in Algorithm 1. It is analogous to the forward-backward algorithm in the HMM. The computational com-

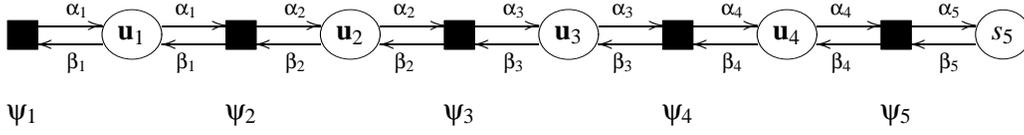


Figure 2: The factor graph corresponding to the change point model and message passing scheme for a model with four observations.

plexity of this algorithm is due to the conditional Gaussian factors and the implied increase in the complexity of the messages.

Algorithm 1 The sum-product algorithm for the SLDS

Forward pass Start the recursion with

$$p(\mathbf{u}_1 | \mathbf{y}_1, \theta) \equiv \alpha_1(\mathbf{u}_1) = \frac{\Psi_1(\mathbf{u}_1)}{Z_1}, \quad Z_1 = \sum_{\mathbf{u}_1} \Psi_1(\mathbf{u}_1).$$

For $t = 1, \dots, T$

$$p(\mathbf{u}_t | \mathbf{y}_{1:t}, \theta) \equiv \alpha_t(\mathbf{u}_t) = \frac{\sum_{\mathbf{u}_{t-1}} \alpha_{t-1}(\mathbf{u}_{t-1}) \Psi_t(\mathbf{u}_{t-1,t})}{Z_{t|t-1}},$$

with $p(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta) \equiv Z_{t|t-1} = \sum_{\mathbf{u}_{t-1,t}} \alpha_{t-1}(\mathbf{u}_{t-1,t}) \Psi_t(\mathbf{u}_{t-1,t})$.

Backward pass If s_{T+1} is not observed, start the recursion with

$$\beta_T(\mathbf{u}_T) = 1.$$

If s_{T+1} is observed the definition of $\beta_T(\mathbf{u}_T)$ is changed accordingly: if $s_{T+1} = n$, then $\beta_T(s_T = p) = 0$. Similarly if $s_{T+1} = p$ then $\beta_T(s_T = n) = 0$.

For $t = T - 1, T - 2, \dots, 1$

$$\frac{p(\mathbf{y}_{t+1:T} | \mathbf{u}_t, \theta)}{p(\mathbf{y}_{t+1:T} | \mathbf{y}_{1:t}, \theta)} \equiv \beta_t(\mathbf{u}_t) = \frac{\sum_{\mathbf{u}_{t+1}} \Psi_{t+1}(\mathbf{u}_{t,t+1}) \beta_{t+1}(\mathbf{u}_{t+1})}{\prod_{v=t+1}^T Z_{v|v-1}}.$$

After a forward-backward pass, single-slice and two-slice posteriors are given by

$$\begin{aligned} p(\mathbf{u}_t | \mathbf{y}_{1:T}, \theta) &= \alpha_t(\mathbf{u}_t) \beta_t(\mathbf{u}_t) \\ p(\mathbf{u}_{t-1,t} | \mathbf{y}_{1:T}, \theta) &= \frac{1}{Z_{t|t-1}} \alpha_{t-1}(\mathbf{u}_{t-1,t}) \Psi_t(\mathbf{u}_{t-1,t}) \beta_t(\mathbf{u}_t). \end{aligned}$$

In the forward pass the message $\alpha_t(\mathbf{u}_t) \equiv p(\mathbf{x}_t, s_t | \mathbf{y}_{1:t}, \theta)$ is *not* conditional Gaussian, but a mixture of Gaussians conditioned on the regime indicator s_t . It has t components in total: conditioned on $s_t = n$ the posterior contributes a single Gaussian component $p(\mathbf{x}_t | s_t = n, \mathbf{y}_{1:t}, \theta)$ conditioned

on $s_t = p$ the posterior $p(\mathbf{x}_t | s_t = p, \mathbf{y}_{1:t}, \theta)$ is a mixture of Gaussians with $t - 1$ components: each component corresponds to a possible starting point of the predefault regime.

In the smoothing pass an analogous growth of the number of components in the backward messages $\beta_t(\mathbf{u}_t)$ occurs, but now growing backwards in time. The single-slice posterior, which is obtained from the product of the forward and backward messages, has T components for all t .

Note that the linear complexity in T is special for the change point model with the restricted regime transitions. In general the number of components in the posterior would grow exponentially.

4. MAP Parameter Estimation

In Section 3 we have assumed that the model parameters θ were known. If they are not known, the expectation-maximization (EM) algorithm (Dempster et al., 1977) with Algorithm 1 in the expectation step, can be used to find maximum likelihood (ML) or maximum a posteriori (MAP) parameter settings. Appendix B lists the M-step updates for the change point model. Appendix C discusses sensible priors on the transition probabilities in Π .

The learning setting is *semi supervised*. We assume we are given a set of V training sequences $\{\mathbf{y}_{1:T_v}^{(v)}\}_{v=1}^V$ and that for some, possibly all, we observe $s_{T+1}^{(v)}$. All sequences v for which $s_{T+1}^{(v)} = s$ can be used to estimate the parameters of the normal regime. If $s_{T+1}^{(v)} = f$ or not observed, the change point from the normal to the predefault regime is inferred in the E-step. The updates from Appendix B then boil down to weighted variants of the linear dynamical system M-step updates, where the weights correspond to the posterior probabilities of being in a particular regime.

The EM algorithm is guaranteed to converge to a local maximum of the likelihood/parameter posterior. Different initial parameter estimates $\theta^{(0)}$ may lead the algorithm to converge to different local maxima. This is a known property of the EM algorithm for fitting a mixture of Gaussians. In the current model it can be hoped that the dependence on initialization is less than in the general mixtures of Gaussian case. If there are sequences that are known to be entirely normal (when $s_{T+1} = s$) these sequences are only used to determine the characteristics of the normal regime. Also, due to the change point restriction, some ambiguity is resolved since it is known that the normal precedes the predefault regime.

5. Approximate Inference: Kikuchi Free Energies with Weak Consistency Constraints

The exact inference algorithm presented in Section 3 has the same form as the HMM and Kalman filter algorithms. The messages that are sent, $\alpha_t(\mathbf{u}_t)$ and $\beta_t(\mathbf{u}_t)$, are not in the conditional Gaussian family, but are conditional mixtures. As was discussed in Section 3, the number of components in the mixtures grows linear with t and $T - t$ respectively.

A straightforward approximation is to approximate these messages by a conditional Gaussian in every step. This implies that every message stores only two components, regardless of t . In the forward pass the best approximating conditional Gaussian can be defined in Kullback-Leibler (KL) sense. The approximating conditional Gaussian is then found by *moment matching* or a *collapse* (see Appendix A). The oldest use of this approach we are aware of is in Harrison and Stevens (1976).

A symmetric approximation for the backward pass working directly on the $\beta_t(\mathbf{u}_t) = \frac{p(\mathbf{y}_{t+1:T}|\mathbf{u}_t, \theta)}{p(\mathbf{y}_{t+1:T}|\mathbf{y}_{1:t}, \theta)}$ messages cannot be formulated, since in contrast to the α messages, the β messages in general will not be proper distributions and hence a KL divergence is not defined.

This has led to other approaches that introduced additional approximations beyond the projection onto the conditional Gaussian family, (e.g. Shumway and Stoffer, 1991; Kim, 1994). The expectation propagation (EP) framework of Minka (2001) is very suited for this particular model and essentially formulates a backward pass symmetric to the approach outlined above (Zoeter and Heskes, 2005). There are at least two ways of looking at EP. In the first, EP is seen as an iteration scheme where at every step an exact model potential is added to the approximation followed by a projection onto a chosen approximating family. In the second, EP is derived from a particular variational problem. The EP algorithm is introduced in Section 5.2 using the second point of view, which facilitates the description of our generalization in Section 5.3. For a presentation of EP as an iteration of projections the reader is referred to Minka (2001).

The approximate filter and the EP algorithm share that they are greedy: the approximations are made locally. In the EP algorithm the local approximations are made as consistent as possible by iteration. There is no guarantee that the resulting means and covariances in the conditional Gaussian families equal the means and covariances of the exact posteriors. The strong junction tree framework of Lauritzen (1992) operates on trees with larger cliques and approximates messages on a global level. Thereby it *does* guarantee exactness of means and covariances. For the SLDS a strong junction tree has at least one cluster that effectively contains all discrete variables.

Section 5.3 introduces a generalization of the EP algorithm from Section 5.2. In the generalization, an extra integer parameter κ is introduced that allows a trade-off between computation time and accuracy. The EP algorithm from Zoeter and Heskes (2005) and the strong junction tree from Lauritzen (1992) are then on both extremes.

5.1 Exact Inference as an Energy Minimization Procedure

To facilitate the introduction of the expectation and the generalized expectation propagation algorithms, exact inference is introduced in this Section as a minimization procedure. Expectation propagation will follow from an approximation of the objective.

We start by following the variational approach (e.g. Jaakkola, 2001) and turn the computation of $-\log Z \equiv -\log p(\mathbf{y}_{1:T}|\theta)$ into an optimization problem:

$$-\log Z = \min_{\tilde{p}} [-\log Z + \text{KL}(\tilde{p}(\mathbf{u}_{1:T})||p(\mathbf{u}_{1:T}|\mathbf{y}_{1:T}, \theta))] \quad (2)$$

$$= \min_{\tilde{p}} \left[-\log Z + \sum_{\mathbf{u}_{1:T}} \tilde{p}(\mathbf{u}_{1:T}) \log \frac{\tilde{p}(\mathbf{u}_{1:T})}{Z^{-1} \prod_{t=1}^T \psi_t(\mathbf{u}_{t-1,t})} \right] \quad (3)$$

$$= \min_{\tilde{p}} \left[-\sum_{t=1}^T \sum_{\mathbf{u}_{t-1,t}} \tilde{p}(\mathbf{u}_{t-1,t}) \log \psi_t(\mathbf{u}_{t-1,t}) + \sum_{\mathbf{u}_{1:T}} \tilde{p}(\mathbf{u}_{1:T}) \log \tilde{p}(\mathbf{u}_{1:T}) \right]. \quad (4)$$

In (2)–(4) the minimization is over all valid distributions $\tilde{p}(\mathbf{u}_{1:T})$ on the domain $\mathbf{u}_{1:T}$. The KL term in (2) is guaranteed to be positive and equals zero if and only if $\tilde{p}(\mathbf{u}_{1:T}) = p(\mathbf{u}_{1:T}|\mathbf{y}_{1:T}, \theta)$ (Gibbs inequality). This guarantees the equality in (2).

In terms of \mathbf{u}_t the exact posterior factors as

$$p(\mathbf{u}_{1:T} | \mathbf{y}_{1:T}, \theta) = \frac{\prod_{t=2}^T p(\mathbf{u}_{t-1,t} | \mathbf{y}_{1:T}, \theta)}{\prod_{t=2}^{T-1} p(\mathbf{u}_t | \mathbf{y}_{1:T}, \theta)}, \quad (5)$$

so we can restrict the minimization in (4) to be over all valid distributions of the form (5):

$$-\log Z = \min_{\{\tilde{p}_t, \tilde{q}_t\}} \left[- \sum_{t=2}^T \sum_{\mathbf{u}_{t-1,t}} \tilde{p}_t(\mathbf{u}_{t-1,t}) \log \Psi_t(\mathbf{u}_{t-1,t}) + \sum_{t=2}^T \sum_{\mathbf{u}_{t-1,t}} \tilde{p}_t(\mathbf{u}_{t-1,t}) \log \tilde{p}_t(\mathbf{u}_{t-1,t}) - \sum_{t=2}^{T-1} \sum_{\mathbf{u}_t} \tilde{q}_t(\mathbf{u}_t) \log \tilde{q}_t(\mathbf{u}_t) \right]. \quad (6)$$

The minimization is now with respect to one-slice beliefs $\tilde{q}_t(\mathbf{u}_t)$ and two-slice beliefs $\tilde{p}_t(\mathbf{u}_{t-1,t})$ under the constraints that these beliefs are properly normalized and consistent:

$$\tilde{p}_t(\mathbf{u}_t) = \tilde{q}_t(\mathbf{u}_t) = \tilde{p}_{t+1}(\mathbf{u}_t). \quad (7)$$

To emphasize that the above constraints are exact, and to distinguish them from the *weak consistency constraints* that will be introduced below, we will refer to (7) as *strong consistency constraints*.

Minimizing the objective in (6) under normalization and strong consistency constraints (7) gives exact one- and two-slice posteriors. Since they are exact, the one-slice beliefs $\tilde{q}_t(\mathbf{u}_t)$ will have T components in our change point model and M^T components in a general SLDS.

5.2 Expectation Propagation

As we have seen in the previous section, exact inference can be interpreted as a minimization procedure under constraints. At the minimum, the variational parameters $\tilde{q}_t(\mathbf{u}_t)$ are equal to the exact single node marginals. Since these marginals have many components (T in our changepoint model, M^T in a general SLDS) even storing the results is computationally demanding.

To obtain an approximation the variational parameters $\tilde{q}_t(\mathbf{u}_t)$ are restricted to be conditional Gaussian. Recall that $\mathbf{u}_t \equiv \{s_t, \mathbf{x}_t\}$, so that the conditional Gaussian restriction implies that for every possible value for s_t , \mathbf{x}_t follows a Gaussian distribution, instead of a mixture of Gaussians with a mixture component for every possible regime history for $s_{1:t-1,t+1:T}$. This restriction is analogous to the approximation in the generalized pseudo Bayes 2 (GPB 2) filter (Bar-Shalom and Li, 1993) where in every time update step mixtures of Gaussians are collapsed onto single Gaussians. In fact, as we will see shortly, GPB2 can be seen as a first forward pass in the algorithm that follows from our current approach.

The conditional Gaussian form of $\Psi_t(\mathbf{u}_{t-1,t})$ and the conditional Gaussian choice for $\tilde{q}_t(\mathbf{u}_t)$ imply that at the minimum in (6) $\tilde{p}_t(\mathbf{u}_{t-1,t})$ is conditionally Gaussian as well (see Appendix D).

If we restrict the form of $\tilde{q}_t(\mathbf{u}_t)$, but leave the consistency constraints exact as in (7), a minimum of the free energy has a very restricted form. The strong consistency constraints would imply that the two exact marginals $\sum_{\mathbf{u}_{t-1}} \tilde{p}_t(\mathbf{u}_{t-1,t}) = \tilde{p}_t(\mathbf{u}_t)$ and $\sum_{\mathbf{u}_t} \tilde{p}_t(\mathbf{u}_{t-1,t}) = \tilde{p}_{t-1}(\mathbf{u}_{t-1})$ are conditional Gaussians instead of conditional mixtures. This holds only if the continuous variables $x_{t-1,t}$ are independent of the discrete states $s_{t-1,t}$ in $\tilde{p}_t(\mathbf{u}_{t-1,t})$.

To obtain non-trivial approximations, the single-slice beliefs $\tilde{q}_t(\mathbf{u}_t)$ are restricted to be conditional Gaussian as outlined above, and in addition the consistency constraints are weakened. Instead

of having equal marginals we only require overlapping beliefs to be consistent on their overlapping expectations

$$\langle f(\mathbf{u}_t) \rangle_{\tilde{p}_t} = \langle f(\mathbf{u}_t) \rangle_{\tilde{q}_t} = \langle f(\mathbf{u}_t) \rangle_{\tilde{p}_{t+1}}, \quad (8)$$

where $f(\mathbf{u}_t)$ is the vector of sufficient statistics of the conditional Gaussian family over \mathbf{u}_t as defined in Appendix A.

With these restrictions $\tilde{q}_t(\mathbf{u}_t)$ is in general not the marginal of $\tilde{p}_t(\mathbf{u}_{t-1,t})$, so one-slice and two-slice beliefs satisfying (8) do not lead to a proper distribution of the form (5). As a result, although we started the derivation with the variational (mean-field) bound (2), the objective we aim to minimize is not guaranteed to be a bound on $-\log Z$.

The EP algorithm can be seen as fixed point iteration in the *dual space* of the constrained minimization problem (Zoeter and Heskes, 2005, Appendix D). This is in direct analogy to the interpretation of loopy belief propagation as fixed point iteration in the dual space of the Bethe free energy (Yedidia et al., 2005).

Algorithm 2 presents the generalization that will be derived next, but with $\kappa = 0$ it gives the basic update equations of this section. In a first forward pass, with all backward messages initialized as $\beta_t(\mathbf{u}_t) = 1$ (i.e. effectively with no backward messages), the updates are equivalent to the greedy projection filter GPB2.

As a final note we remark that this approximation, and even the update scheme, can also be derived from the iterative projection point of view of EP. To obtain Algorithm 2 with $\kappa = 0$, the approximating family should be chosen to be a product of independent conditional Gaussians (Zoeter and Heskes, 2005).

5.3 Generalized Expectation Propagation

Since we have associated the EP approach to an approximation of the Bethe free energy (6), we can extend the approximation analogously to Kikuchi's extension of the Bethe free energy (Yedidia et al., 2005).

In the EP free energy (6) the minimization is w.r.t. beliefs over *outer clusters*, $\tilde{p}_t(\mathbf{u}_{t-1,t})$, and their *overlaps*, $\tilde{q}_t(\mathbf{u}_{t-1,t})$. In the so-called *negative entropy*,

$$\sum_{t=2}^T \sum_{\mathbf{u}_{t-1,t}} \tilde{p}_t(\mathbf{u}_{t-1,t}) \log \tilde{p}_t(\mathbf{u}_{t-1,t}) - \sum_{t=2}^{T-1} \sum_{\mathbf{u}_t} \tilde{q}_t(\mathbf{u}_t) \log \tilde{q}_t(\mathbf{u}_t),$$

from (6), the outer clusters enter with a plus, the overlaps with a minus sign. These 1 and -1 factors can be interpreted as *counting numbers* that ensure that every variable effectively is counted once in the (approximate) entropy in (6). If the free energy is exact (i.e. no parametric choice for the beliefs, and strong consistency constraints), the local beliefs are exact marginals, and as in (5), the counting numbers can be interpreted as powers that dictate how to construct a global distribution from the marginals.

In Kikuchi's extension the outer clusters are taken larger. The minimization is then w.r.t. beliefs over outer clusters, their direct overlaps, the overlaps of the overlaps, etc. With each belief again proper counting numbers are associated.

One way to construct a valid Kikuchi based approximation is as follows (Yedidia et al., 2005). Choose outer clusters $\mathbf{u}_{outer(i)}$ and associate with them the counting number $c_{outer(i)} = 1$. The outer clusters should be such that all domains $\mathbf{u}_{t-1,t}$ of the model potentials $\Psi_t(\mathbf{u}_{t-1,t})$ are fully contained

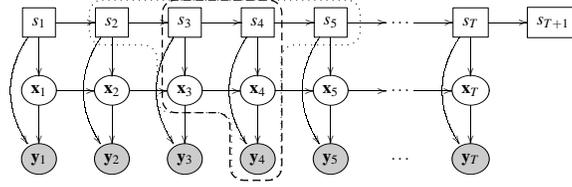


Figure 3: Cluster definitions for $\kappa = 0$ (dashed) and $\kappa = 1$ (dotted).

in at least one outer cluster. Then recursively define the overlaps of the outer clusters $\mathbf{u}_{over(i)}$, the overlaps of the overlaps, etc. The counting number associated with cluster γ is given by the Möbius recursion

$$c_\gamma = 1 - \sum_{\mathbf{u}_\gamma \supset \mathbf{u}_\gamma} c_\gamma. \quad (9)$$

A crucial observation for the SLDS is that it makes sense to take outer clusters larger than the cliques of a (weak) junction tree. If we do not restrict the parametric form of $\tilde{q}_t(\mathbf{u}_t)$ and keep exact constraints, the cluster choice in (5) gives exact results. However, the restriction that $\tilde{q}_t(\mathbf{u}_t)$ must be conditional Gaussian, and the weak consistency constraints imply an approximation: only part of the information from the past can be passed on to the future and vice versa. With weak constraints it is beneficial to take larger outer clusters and larger overlaps, since the weak consistency constraints are then over a larger set of sufficient statistics and hence “stronger”.

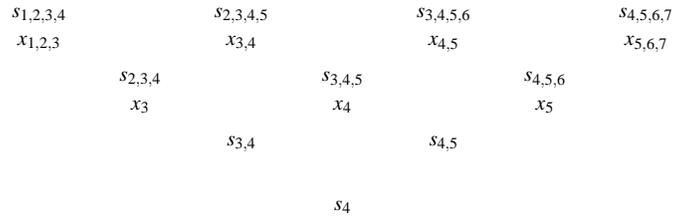
We define symmetric extensions of the outer clusters as depicted in Figure 3. The size of the clusters is indicated by $0 \leq \kappa \leq \lceil \frac{T-2}{2} \rceil$:

$$\mathbf{u}_{outer(i)} = \{s_{i:i+2(\kappa+1)-1}, x_{i+\kappa, i+\kappa+1}\}, \quad \text{for } i > 1 \wedge i < T - 2\tau + 2 \quad (10)$$

$$\mathbf{u}_{over(i)} = \mathbf{u}_{outer(i)} \cap \mathbf{u}_{outer(i+1)}. \quad (11)$$

In the outer clusters only the discrete space is extended because the continuous part can be integrated out analytically and the result stays in the conditional Gaussian family. The first and the last outer cluster have a slightly larger set. In addition to the set (10) the first cluster also contains $\mathbf{x}_{1:i+\kappa-1}$ and the last also $\mathbf{x}_{i+\kappa+2:T}$. This implies a choice where the number of outer clusters is as small as possible at the cost of a larger continuous part in the first and the last cluster. A slightly different choice would have more clusters, but only two continuous variables in every outer cluster.

To demonstrate the construction of clusters and the computation of their associated counting numbers we will look at the case of $\kappa = 1$. Below the clusters are shown schematically, with outer clusters on the top row, and recursively the overlaps of overlaps, etc.



The outer clusters all have counting number 1. The direct overlaps each have two larger clusters in which they are contained. Their associated counting numbers follow from (9) as $1 - 2 = -1$.

The overlaps of overlaps have five clusters in which they are contained, their counting numbers are $1 - (3 - 2) = 0$. The clusters on the lowest level have nine parents, which results in a counting number $1 - (4 - 3 + 0) = 0$. It is easily verified that with $\kappa = 0$ we obtain the cluster and counting number choice of Section 5.2.

A second crucial observation for the SLDS is that the choice of outer clusters (10) implies that we only have to consider outer clusters and direct overlaps, i.e. the phenomenon that all clusters beyond the direct overlaps get an associated counting number of 0 in the example above extends to all κ . This is a direct result of the fact that the clusters from (10) form the cliques and separators in a (weak) junction tree. I.e. another way to motivate a generalization with the cluster choice (10) is to replace (5) with

$$p(\mathbf{u}_{1:T} | \mathbf{y}_{1:T}, \theta) = \frac{\prod_{i=1}^N p(\mathbf{u}_{outer(i)} | \mathbf{y}_{1:T}, \theta)}{\prod_{j=1}^{N-1} p(\mathbf{u}_{over(j)} | \mathbf{y}_{1:T}, \theta)}, \quad (12)$$

and use this choice in (4) to obtain an extension of (6). In (12), $N = T - 2\kappa - 1$ denotes the number of outer clusters in the approximation.

The aim then becomes to minimize

$$\begin{aligned} \mathcal{F}_{GEP} = & - \sum_{i=1}^N \sum_{\mathbf{u}_{outer(i)}} \tilde{p}_i(\mathbf{u}_{outer(i)}) \log \Psi^{(i)}(\mathbf{u}_{outer(i)}) \\ & + \sum_{i=1}^N \sum_{\mathbf{u}_{outer(i)}} \tilde{p}_i(\mathbf{u}_{outer(i)}) \log \tilde{p}_i(\mathbf{u}_{outer(i)}) \\ & - \sum_{i=1}^{N-1} \sum_{\mathbf{u}_{over(i)}} \tilde{q}_i(\mathbf{u}_{over(i)}) \log \tilde{q}_i(\mathbf{u}_{over(i)}), \end{aligned} \quad (13)$$

w.r.t. the potentials $\tilde{p}_i(\mathbf{u}_{outer(i)})$, and $\tilde{q}_i(\mathbf{u}_{over(i)})$. For $i = 2, 3, \dots, N-1$, the potentials $\Psi^{(i)}(\mathbf{u}_{over(i)})$ are identical to the potentials $\psi_{i+\kappa+1}(\mathbf{u}_{i+\kappa, i+\kappa+1})$ from (1). At the boundaries they are a product of potentials that are ‘‘left over’’:

$$\begin{aligned} \Psi^{(1)} &= \prod_{j=1}^{\kappa+2} \psi_j(\mathbf{u}_{j-1, j}) \\ \Psi^{(N)} &= \prod_{j=T-\kappa}^T \psi_j(\mathbf{u}_{j-1, j}), \end{aligned}$$

with $\Psi^{(1)} = \prod_{j=1}^T \psi_j(\mathbf{u}_{j-1, j})$ if $N = 1$.

The approximation in the *generalized EP free energy*, \mathcal{F}_{GEP} , arises from the restriction that $\tilde{q}_i(\mathbf{u}_{over(i)})$ is conditional Gaussian and from the fact that overlapping potentials are only required to be weakly consistent

$$\langle f(\mathbf{u}_{over(i)}) \rangle_{\tilde{p}_i} = \langle f(\mathbf{u}_{over(i)}) \rangle_{\tilde{q}_i} = \langle f(\mathbf{u}_{over(i)}) \rangle_{\tilde{p}_{i+1}}.$$

The benefit of the (weak) junction tree choice of outer clusters and overlaps is that we can employ the same algorithm for the $\kappa = 0$ as for the $\kappa > 0$ case. Algorithm 2 can be seen as a single-loop minimization heuristic. As mentioned above, and as shown in Appendix D, the algorithm can be interpreted as fixed point iteration in the space of Lagrange multipliers that are added to (13) to

enforce the weak consistency constraints. Just as for EP itself, convergence of Algorithm 2 is not guaranteed.

In Algorithm 2 the messages are initialized as conditional Gaussian potentials, such that

$$\tilde{q}(\mathbf{u}_{\text{over}(i)}) = \alpha_i(\mathbf{u}_{\text{over}(i)})\beta_i(\mathbf{u}_{\text{over}(i)})$$

are normalized. A straightforward initialization would be to initialize all messages with 1. If at the start all products of matching messages are normalized, we can interpret the product of local normalizations $\prod_{i=1}^N Z_i$ as an approximation of the normalization constant Z .

Algorithm 2 Generalized EP for an SLDS

Compute a forward pass by performing the following steps for $i = 1, 2, \dots, N-1$, with $i' \equiv i$, and a backward pass by performing the same steps for $i = N, N-1, \dots, 2$, with $i' \equiv i-1$. Iterate forward-backward passes until convergence. At the boundaries keep $\alpha_0 = \beta_N = 1$.

1. Construct an outer-cluster belief,

$$\tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) = \frac{\alpha_{i-1}(\mathbf{u}_{\text{over}(i-1)})\Psi^{(i)}(\mathbf{u}_{\text{outer}(i)})\beta_i(\mathbf{u}_{\text{over}(i)})}{Z_i},$$

$$\text{with } Z_i = \sum_{\mathbf{u}_{\text{outer}(i)}} \alpha_{i-1}(\mathbf{u}_{\text{over}(i-1)})\Psi^{(i)}(\mathbf{u}_{\text{outer}(i)})\beta_i(\mathbf{u}_{\text{over}(i)}).$$

2. Marginalize to obtain a one-slice marginal

$$\tilde{p}_i(\mathbf{u}_{\text{over}(i')}) = \sum_{\mathbf{u}_{\text{outer}(i)} \setminus \mathbf{u}_{\text{over}(i')}} \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}).$$

3. Find $\tilde{q}_{i'}(\mathbf{u}_{\text{over}(i')})$ that approximates $\tilde{p}_i(\mathbf{u}_{\text{over}(i')})$ best in Kullback-Leibler (KL) sense:

$$\tilde{q}_{i'}(\mathbf{u}_{\text{over}(i')}) = \text{Collapse}(\tilde{p}_i(\mathbf{u}_{\text{over}(i')})) .$$

4. Infer the new message by division.

$$\alpha_i(\mathbf{u}_{\text{over}(i)}) = \frac{\tilde{q}_i(\mathbf{u}_{\text{over}(i)})}{\beta_i(\mathbf{u}_{\text{over}(i)})}, \quad \beta_{i-1}(\mathbf{u}_{\text{over}(i-1)}) = \frac{\tilde{q}_{i-1}(\mathbf{u}_{\text{over}(i-1)})}{\alpha_{i-1}(\mathbf{u}_{\text{over}(i-1)})}.$$

Figure 4 gives a graphical representation of Algorithm 2 for $\kappa = 0$. Figure 5 gives a similar schema for $\kappa = 1$. The two figures show what information is lost when the one-slice beliefs are collapsed.

The choice of $0 \leq \kappa \leq \lceil \frac{T-2}{2} \rceil$ now allows a trade off between computational complexity and degrees of freedom in the approximation. With $\kappa = 0$, we obtain the EP/Bethe free energy equivalent to Zoeter and Heskes (2005). With $\kappa = \lceil \frac{T-2}{2} \rceil$ there is only one cluster and we obtain a *strong* junction tree, and the found posteriors are exact. Just as with the Kikuchi extension of belief propagation, there is no guaranteed monotonic improvement for intermediate κ 's (Kappen and Wiegierinck, 2002). However, in the change point model, where there are no loops and larger clusters only imply more statistics being propagated between time-slices, we expect improvements

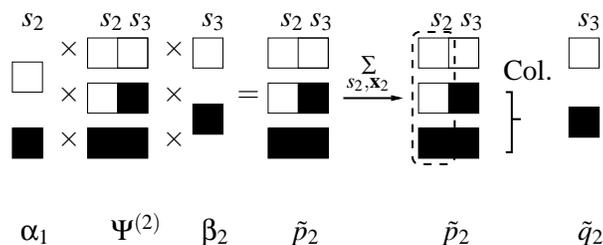


Figure 4: A schematic representation of steps 1, 2 and 3 from Algorithm 2 with $\kappa = 0$, for a sequence with more than 3 observations. The potential $\Psi^{(2)}(\mathbf{u}_{2,3})$ contains three Gaussian components: $p(\mathbf{y}_3, \mathbf{x}_3 | \mathbf{x}_2, s_2 = n, s_3 = n)$, $p(\mathbf{y}_3, \mathbf{x}_3 | \mathbf{x}_2, s_2 = n, s_3 = p)$, and $p(\mathbf{y}_3, \mathbf{x}_3 | \mathbf{x}_2, s_2 = p, s_3 = p)$. The (p, n) assignment gets zero weight by the non-recovery assumption and is therefore not shown. Combinations with absorbing states are excluded since the sequence does not stop at 3. Every component is encoded by a row, with white squares denoting normal, and black squares prefaul regimes. The messages $\alpha_1(\mathbf{x}_2, s_2)$ and $\beta_2(\mathbf{x}_3, s_3)$ are conditional Gaussian by construction and hence each have two components: one corresponding to normal and one to prefaul. Exact marginalization gives $\tilde{p}_2(\mathbf{x}_3, s_3)$, which still consists of three components. To emphasize that s_2 is not part of the domain, it is enclosed by a dashed rectangle. Conditioned on $s_3 = p$, $\tilde{p}_2(\mathbf{x}_3 | s_3 = p)$ is a mixture. This mixture is collapsed to obtain a conditional Gaussian approximation $\tilde{q}_2(\mathbf{u}_3)$.

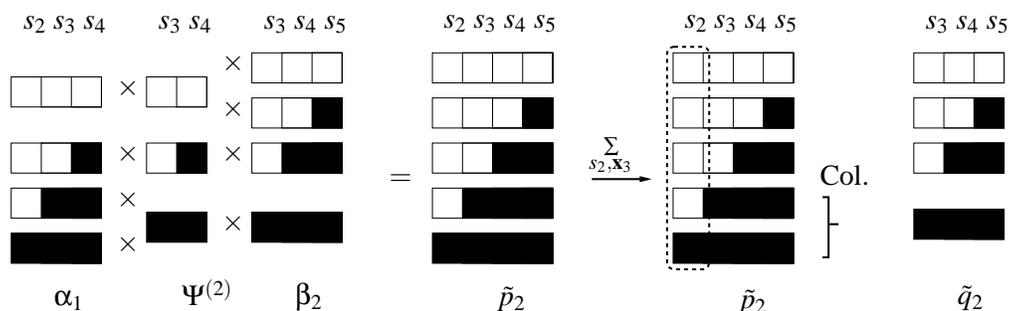


Figure 5: The steps in Algorithm 2 with $\kappa = 1$ are analogous to the steps with $\kappa = 0$ as depicted in Figure 4. With $\kappa = 1$ the two components that are approximated are expected to be very similar: they have been updated with the same transition and observation models in the last three time-slices.

to be extremely likely. In fact, we have not seen the performance degrade with larger κ in any of our experiments.

6. Experiments

In Section 6.1 we explore the properties of the learning algorithm and approximate inference in a controlled setting with artificial data. Section 6.2 presents experiments with EMG data.

6.1 Synthetic Data

As discussed in Section 4 the constraints in the regime transitions aid in learning. When a stop is observed in the trainset, the entire sequence is guaranteed to be normal. Also, the fact that the normal regime precedes the pre-fault regime resolves the invariance under relabeling that would be present in a general switching linear dynamical systems setting. Experiments with artificially generated data shows that even with a relatively small trainset the two regimes can be learned fairly reliably.

We ran experiments where 15 train and 5 test sequences were generated from randomly drawn change point models. The classes of the train sequences (stop or fault) were observed, the classes of the test sequences were unknown. Figure 6 is not an atypical result. In many experiments we find that both the classification (determining whether the sequence ended in a stop or in a fault) and the determination of the change point were often (near) perfect.

The MAP

$$\tau_{\text{map}} = \underset{\tau}{\operatorname{argmax}} p(s_{1:\tau} = n, s_{\tau+1:T} = p | \mathbf{y}_{1:T}, \theta_{\text{ML}}),$$

is taken as the predicted change point. In 10 replications the mean squared error between the actual and the inferred change point was 6.6 (standard deviation 11.75, median 0).

These results are encouraging, but may also be largely due to the fact that arbitrarily drawn models may not pose a serious challenge. Qualitatively the replicated experiments show that for most replications the errors are close to 0 (as in Figure 6). This explains the low median. In a few replications the model has learned normal and pre-fault classes that are different from the true generating model and hence result in large errors. In these replications we still see the “arbitrariness” of the fitted clusters that is common to the mixtures of Gaussians learning. We do not investigate a proper characterization of “difficult” and “easy” models here, but discuss some of the possible pitfalls with the approach in Section 6.2.

To explore the properties of the approximations developed in Section 5, we ran 10 experiments where a single sequence of length 10 was generated from a randomly drawn model. For every sequence, approximate single node posteriors $\tilde{q}(\mathbf{x}_t | \mathbf{y}_{1:T}, \theta)$ were computed using Algorithm 2 with $\kappa = 0, 1, 2, 3, 4$. Figure 7 shows the maximum absolute error in the single node posterior means as a function of κ . The lines show the average over the 10 experiments, the maximum encountered, and the minimum. For sequences with length 10, $\kappa = 4$ is guaranteed to give exact results. So in theory, the lines in Figure 7 should meet at $\kappa = 4$. The discrepancies in Figure 7 are explained by different round off errors in our implementations of Algorithm 2 and the strong junction tree.

As expected the approximations are very good and improve with the size of κ . It must be emphasized however, that the improvement with larger κ can be expected based on intuition, but is not guaranteed.

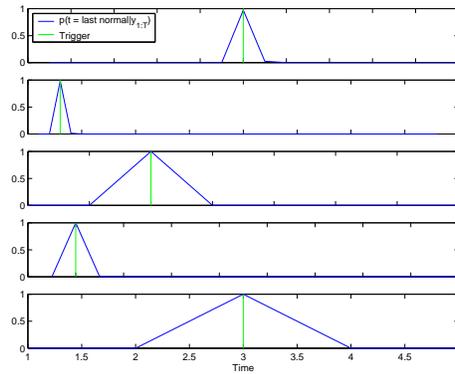


Figure 6: Shown are the inferred and true change points on 5 test sequences. The EM algorithm from Section 4 was presented with 15 artificially generated train sequences, all of which resulted in an observed fault.

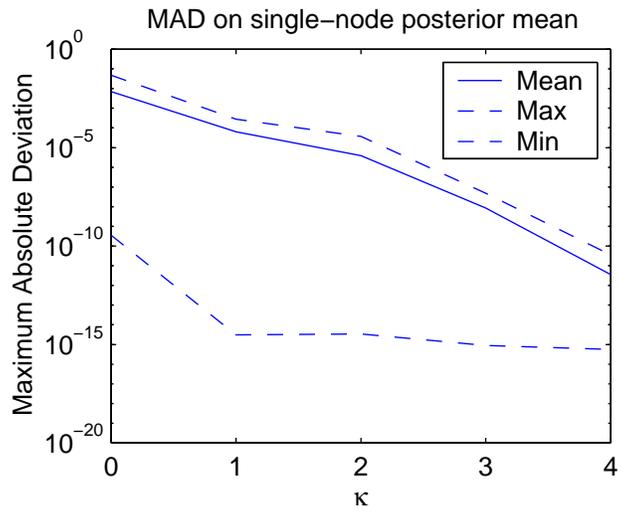


Figure 7: Maximum absolute deviation between exact and approximate single-slice posterior state mean as a function of κ . Shown are the mean, maximum and minimum over ten replications. In all replications $T=10$, so $\kappa = 4$ gives exact results. The small differences between the mean, maximum and minimum deviations that are observed in the plot for $\kappa = 4$ are caused by different round off errors in the generalized EP and the original strong junction tree implementations.

6.2 Detecting Changes in EMG Signals

The algorithm from Section 4 was used to detect changes in EMG patterns in the stumbling experiments from Schillings et al. (1996).

In Schillings et al. (1996) bipolar electromyography (EMG) activity in human subjects were recorded. The subjects were walking on a treadmill at 4 km/h. By releasing an object suspended from an electromagnet the subjects could be tripped at a specified phase in the walking pattern. Partially obscured glasses and earplugs ensured that the tripping was unexpected.

In the experiments video recordings and a pressure-sensitive strip attached to the obstacle signaled the tripping onset. We extracted an interesting change point problem from this experiment by only looking at the EMG signals measured at the biceps femoris at the contra lateral side, i.e. by only looking at a signal which is indicative of the activity of the large muscle in the upper leg at the non-obstructed side.

In our experiments we used data for a single subject. The dataset consisted of 15 control trials where no object was released and 8 stumbling experiments. All sequences were of equal length, and started roughly at the same phase in the walking pattern. The control trials were treated as normal sequences and the 8 others as fault sequences. In the first experiments the class of the sequences were assumed to be known and the aim for the algorithm was to determine the change points.

The original series were raised to a power of $-.2$ to obtain signals that seemed in agreement with the additive noise assumptions. The initial parameter settings for the normal regime was in *Fourier form* (West and Harrison, 1997). The chosen harmonic components were obtained from a discrete Fourier transform. Based on the residuals of the 15 normal sequences a model with 4 harmonics was selected.

There were three different phases of training. In the first, only the normal sequences were considered and the transition matrix A_n was kept fixed. In the second phase, again only the normal regime was considered, but A_n was also fitted. In all phases of learning κ was set to 50, i.e. inference results were indistinguishable from exact. The result of the first two phases is characterized by the left plot in Figure 8. In the third phase all parameters were fitted. The pre-fault model was initialized as an outlier model, i.e. the parameters for the pre-fault regime were copies of the normal regime, but the noise covariances were larger. The characteristics of the entire model are depicted in the right plot of Figure 8.

After convergence, the mean absolute distance between the MAP change point and the triggers in the 8 fault sequences is 4.25, with standard deviation 1.49. Figure 9 shows the posteriors $p(s_{1:T} = n, s_{t+1:T} = p | \mathbf{y}_{1:T}, \theta)$ and the trigger signals for the 8 fault sequences. There are two typical errors: the inferred change point for a few sequences is several steps too early, for a few it comes too late. Figure 10 gives the characteristics for the second and the third fault sequences. The MAP of the second sequence falls a few time steps after the trigger. From the left plot in Figure 10, we might judge that the actual response in the biceps femoris actually starts close to the inferred point. These characteristics are also visible in the other sequences with ‘late’ inferred change points. On the other hand, the sequences with too early inferred change points (e.g. the right plot in Figure 10), *do* show a weakness of the current setup. The degrees of freedom that are available in the pre-fault submodel are used to also explain outliers at the end of the normal regime. This is likely to be a problem in the model specification; there is nothing to prevent a discontinuity in the expected muscle activity at a change (as can be seen in the right plot in Figure 8 and in the plots in Figure 10). Adapting

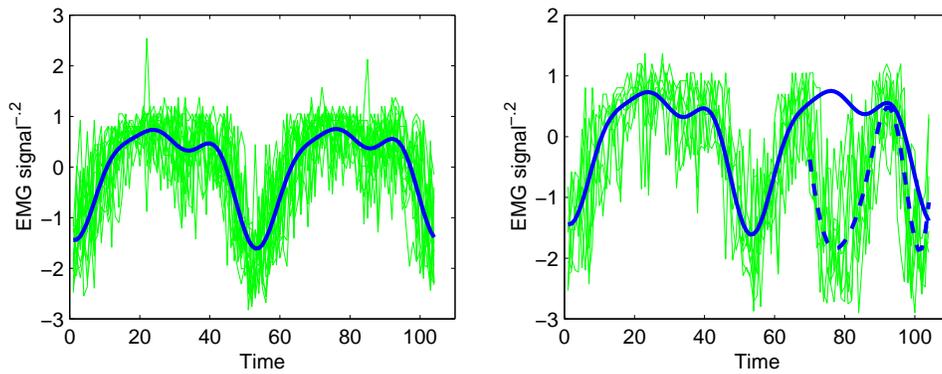


Figure 8: Characteristics of the learned model. The left plot shows the transformed EMG signals for all normal sequences in thin solid lines. The thick solid line shows the model prediction with the regime indicators clamped to normal. The right plot shows all EMG signals from stumbling trials. The light thick line shows model predictions with all regime indicators clamped to normal just as in the left plot. The dark thick line shows the model predictions with the regime indicators clamped to normal from 1 to 70 and to pre-fault from 71 to 104. This change point was hand picked and roughly coincides with the average trigger time.

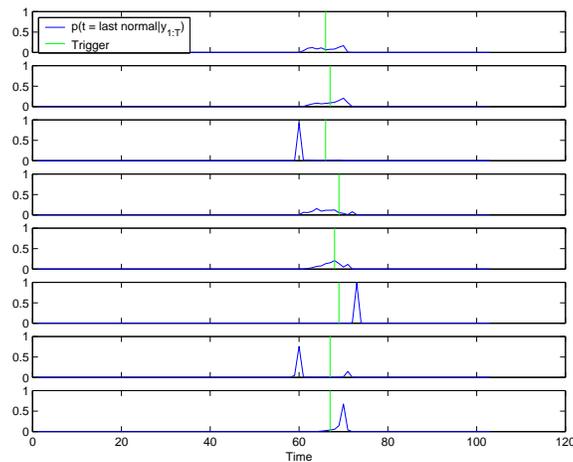


Figure 9: Stumble detection based on a single EMG signal. Vertical bars show the true stumbling trigger. The solid curves show the posterior probability that t was the last normal time point.

the model such that the expected muscle activity is continuous even during a change point might resolve some of the observed overfitting.

To test the classification performance we ran 23 leave-one-out experiments. In every experiment 22 sequences were presented in the training phase. The sequence that was left out was presented after training with s_{T+1} not observed. A simple classification scheme was used; every sequence for

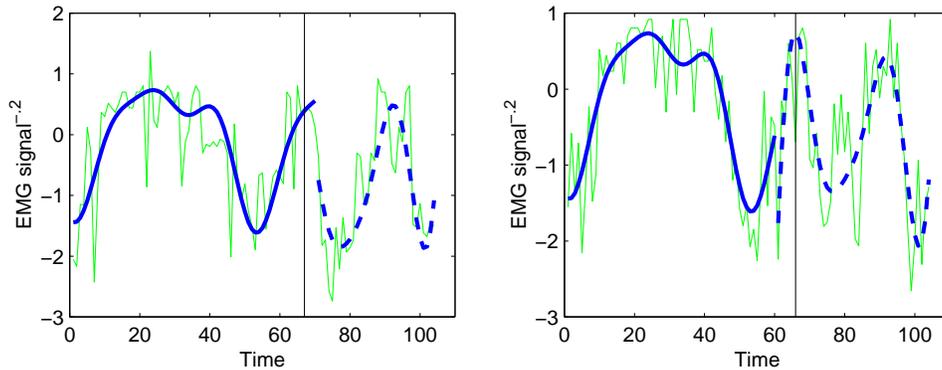


Figure 10: Thin lines show the EMG recordings of a single sequence. The vertical bars show the moment at which the stumble was triggered. The thick lines give an indication of the learned models. They were constructed by clamping the discrete states of the model to the MAP change point value and computing the predicted mean EMG signal (light lines represent the normal, dark lines the prefault regime). The left plot shows the second (from the top) sequence from Figure 9 and gives an acceptable detection of the prefault regime. The right plot shows the third sequence and represents a typical overfit: the model uses its degrees of freedom to fit outliers preceding the change point in some of the sequences. This explains the too early warnings in Figure 9.

which $p(s_{1:T} = n | \mathbf{y}_{1:T}, \theta) \geq .5$ was classified as normal. With this scheme all abnormal sequences, and 13 out of 15 normal sequences were correctly classified.

7. Discussion

Motivated by fault and change detection problems in dynamical systems we have introduced a switching linear dynamical system with constrained regime transition probabilities. The system is assumed to start in a normal regime and to either result in an absorbing stop state or change to a prefault regime. Once the system reaches a prefault regime, it cannot recover and eventually has to result in a fault.

These model assumptions have several advantages. As discussed in Sections 2 and 3, the assumption that the system cannot recover can be exploited to yield an algorithm that computes exact state and regime posteriors in time polynomial in the number of observations.

Another advantage is with learning. An observed stop implies that the system did not change, and an observed fault implies that it did. So if a set of training sequences exists for which the exact change points are unknown, but for which the resulting absorbing states are observed, these model assumptions provide an interesting semi-supervised learning setting. The experiments from Section 6 indicate that these extra assumptions help to solve some of the problems with local minima that occur in general mixtures of Gaussians and SLDS learning. Although overfitting may still occur, careful initialization may be necessary, and violations of the linear Gaussian assumptions may pose problems.

Since the number of observations, T , may grow very large we have introduced an approximate inference algorithm in Section 5.

The algorithm, generalized expectation propagation (GEP), can be derived as a fixed point iteration that aims to minimize a variant of a Kikuchi free energy. One way of interpreting the algorithm is that it sends messages along a weak junction tree as if it was a strong junction tree. This is analogous to the interpretation of loopy belief propagation as an algorithm that sends messages on a loopy graph as if it was operating on a tree.

The change point model has two pleasant properties that makes the application of GEP particularly elegant. The first is the fact that the conditional independencies in the underlying model form a chain. Therefore we can straightforwardly choose outer clusters in the Kikuchi approximation such that they form a (weak) junction tree. We have shown that the resulting GEP updates then simplify since only outer clusters and direct overlaps need to be considered, i.e. from an implementation point of view the algorithm is not more complicated than the ordinary EP algorithm. Also, since there are no loops disregarded, increasing the cluster size leads to relatively “well behaved” approximations; they satisfy the perfect correlation and non-singularity conditions from Welling et al. (2005). Increasing the size of the clusters in our approximation implies that more statistics are passed from past to future and vice versa. This makes an improvement in the approximation very likely (although an improvement is only guaranteed for $\kappa \propto T$ at which point it becomes exact). This is in contrast to the generalization of belief propagation on e.g. complete graphs, which is notorious for the fact that with unfortunate choices of clusters the quality degrades with larger clusters (Kappen and Wiergerinck, 2002). In our experiments with the change point model, we have never observed a degradation of the quality with an increase of κ . This suggests that κ should be set as large as computing power permits.

The first pleasant property of the change point model leads to the observation that in approximations with weak consistency constraints it makes sense to take clusters larger than is necessary to form a (weak) junction tree. This property is shared with all models that have (weak) junction trees with reasonable cluster sizes, in particular chains and trees.

The second property is due to the no-recovery assumption property in the change point model. This implies that exact inference is polynomial in T , and also that approximate inference is polynomial in κ , which makes a wide range of κ 's feasible. In a general SLDS exact inference scales exponential in T and approximate inference exponential in κ .

Although we did not discuss this in Section 5, the GEP algorithm is not restricted to trees or chains. In models with cycles and complicated parametric families, an algorithm can send messages as if it is sending messages on a strong junction tree, whereas the underlying cluster choices do not form a tree, neither a weak nor a strong one. See Heskes and Zoeter (2003) for a discussion.

Algorithm 2 is conjectured to be a proper generalization of the EP framework. Although tree EP (Minka and Qi, 2004) results in approximations that are related to (variants of) Kikuchi free energies it is unlikely that a tree or another clever choice of the approximating family would result in Algorithm 2. Since the overlapping $\tilde{q}(\mathbf{u}_{\text{over}(i)})$ are not strongly consistent they cannot easily be interpreted as marginals of a proper approximating family on which the EP algorithm would project.

Acknowledgments

We would like to thank I. Schillings for providing the EMG data that was used in the experiments of Section 6.2.

Appendix A. Operations on Conditional Gaussian Potentials

To allow for simple notation in the main text this appendix introduces the conditional Gaussian (CG) distribution. A discrete variable s and a continuous variable \mathbf{x} are jointly CG distributed if the marginal of s is multinomial distributed and, conditioned on s , \mathbf{x} is Gaussian distributed. Let \mathbf{x} be d -dimensional and let S be the set of values s can take. In moment form the joint distribution reads

$$p(s, \mathbf{x}) = \pi_s (2\pi)^{-d/2} |\Sigma_s|^{-1/2} \exp \left[-\frac{1}{2} (\mathbf{x} - \mu_s)^\top \Sigma_s^{-1} (\mathbf{x} - \mu_s) \right],$$

with moment parameters $\{\pi_s, \mu_s, \Sigma_s + \mu_s \mu_s^\top\}$, where π_s is positive for all s and satisfies $\sum_s \pi_s = 1$ and Σ_s is a positive definite matrix. The definition of $\Sigma_s + \mu_s \mu_s^\top$ instead of Σ_s is motivated by (16) below. For compact notation sets with elements dependent on s will implicitly ranges over $s \in S$. In canonical form the CG distribution is given by

$$p(s, \mathbf{x}) = \exp \left[g_s + \mathbf{x}^\top \mathbf{h}_s - \frac{1}{2} \mathbf{x}^\top K_s \mathbf{x} \right], \quad (14)$$

with canonical parameters $\{g_s, \mathbf{h}_s, K_s\}$.

The so-called *link function* $g(\cdot)$ maps canonical parameters to moment parameters:

$$\begin{aligned} g(\{g_s, \mathbf{h}_s, K_s\}) &= \{\pi_s, \mu_s, \Sigma_s + \mu_s \mu_s^\top\} \\ \pi_s &= \exp(g_s - \bar{g}) \\ \mu_s &= K_s^{-1} \mathbf{h}_s \\ \Sigma_s &= K_s^{-1}, \end{aligned}$$

with $\bar{g} \equiv \frac{1}{2} \log \left| \frac{K_s}{2\pi} \right| - \frac{1}{2} \mathbf{h}_s^\top K_s \mathbf{h}_s$, the part of g_s that depends on \mathbf{h}_s and K_s . The link function is unique and invertible:

$$\begin{aligned} g^{-1}(\{\pi_s, \mu_s, \Sigma_s + \mu_s \mu_s^\top\}) &= \{g_s, \mathbf{h}_s, K_s\} \\ g_s &= \log \pi_s - \frac{1}{2} \log |2\pi \Sigma_s| - \frac{1}{2} \mu_s^\top \Sigma_s^{-1} \mu_s \\ \mathbf{h}_s &= \Sigma_s^{-1} \mu_s \\ K_s &= \Sigma_s^{-1}. \end{aligned}$$

A conditional Gaussian *potential* is a generalization of the above distribution in the sense that it has the same form as in (14) but need not integrate to 1. K_s is restricted to be symmetric, but need not be positive definite. If K_s is positive definite the moment parameters are determined by $g(\cdot)$. In this section we will use $\phi(s, \mathbf{x}; \{g_s, \mathbf{h}_s, K_s\})$ to denote a CG potential over s and \mathbf{x} with canonical parameters $\{g_s, \mathbf{h}_s, K_s\}$.

Multiplication and division of CG potentials are the straightforward extensions of the analogous operations for multinomial and Gaussian potentials. In canonical form:

$$\begin{aligned}\phi(s, \mathbf{x}; \{g_s, \mathbf{h}_s, K_s\})\phi(s, \mathbf{x}; \{g'_s, \mathbf{h}'_s, K'_s\}) &= \phi(s, \mathbf{x}; \{g_s + g'_s, \mathbf{h}_s + \mathbf{h}'_s, K_s + K'_s\}) \\ \phi(s, \mathbf{x}; \{g_s, \mathbf{h}_s, K_s\})/\phi(s, \mathbf{x}; \{g'_s, \mathbf{h}'_s, K'_s\}) &= \phi(s, \mathbf{x}; \{g_s - g'_s, \mathbf{h}_s - \mathbf{h}'_s, K_s - K'_s\}).\end{aligned}$$

With the above definition of multiplication we can define a unit potential

$$1(s, \mathbf{x}) \equiv \phi(s, \mathbf{x}; \{0, \mathbf{0}, 0\}),$$

which satisfies $1(s, \mathbf{x})p(s, \mathbf{x}) = p(s, \mathbf{x})$ for all CG potentials $p(s, \mathbf{x})$. We will sometimes use the shorthand 1 for the unit potential when its domain is clear from the text.

In a similar spirit we can define multiplication and division of potentials with different domains. If the domain of one of the potentials (the denominator in case of division) forms a subset of the domain of the other we can *extend* the smaller to match the larger and perform a regular multiplication or division as defined above. The continuous domain of the small potential is extended by adding zeros in \mathbf{h}_s and K_s at the corresponding positions. The discrete domain is extended by replicating parameters, e.g. extending s to $[s \ t]^\top$ we use parameters $g_{st} = g_s$, $\mathbf{h}_{st} = \mathbf{h}_s$, and $K_{st} = K_s$.

Marginalization is less straightforward for CG potentials. Integrating out continuous dimensions is analogous to marginalization in Gaussian potentials and is only defined if the corresponding moment parameters are defined. Marginalization is then defined as converting to moment form, ‘selecting’ the appropriate rows and columns from μ_s and Σ_s , and converting back to canonical form. More problematic is the marginalization over discrete dimensions of the CG potential. Summing out s results in a distribution $p(\mathbf{x})$ which is a mixture of Gaussians with mixing weights $p(s)$, i.e. the CG family *is not closed under summation*. In the text we will sometimes use, somewhat sloppily, the \sum notation for both summing out discrete and integrating out continuous dimensions.

We define *weak marginalization* (Lauritzen, 1992), as exact marginalization followed by a *collapse*: a projection of the exact marginal onto the CG family. The projection minimizes the Kullback-Leibler divergence $KL(p||q)$ between p , the exact (strong) marginal and q , the weak marginal:

$$\begin{aligned}q(s, \mathbf{x}) &= \operatorname{argmin}_{q \in CG} KL(p||q) \\ &\equiv \operatorname{argmin}_{q \in CG} \sum_{s, \mathbf{x}} p(s, \mathbf{x}) \log \frac{p(s, \mathbf{x})}{q(s, \mathbf{x})}.\end{aligned}$$

This projection has the property that, conditioned on s the weak marginal has the same mean and covariance as the exact marginal. The weak marginal can be computed by *moment matching* (Whittaker, 1989). If $p(\mathbf{x}|s)$ is a mixture of Gaussians for every s with mixture weights $\pi_{r|s}$, means μ_{sr} , and covariances Σ_{sr} (e.g. the exact marginal $\sum_r p(s, r, \mathbf{x})$ of CG distribution $p(s, r, \mathbf{x})$), the moment matching procedure is defined as

$$\begin{aligned}\text{Collapse}(p(s, \mathbf{x})) &\equiv p(s)\mathcal{N}(\mathbf{x}; \mu_s, \Sigma_s) \\ \mu_s &\equiv \sum_r \pi_{r|s} \mu_{sr} \\ \Sigma_s &\equiv \sum_r \pi_{r|s} \left(\Sigma_{sr} + (\mu_{sr} - \mu_s)(\mu_{sr} - \mu_s)^\top \right).\end{aligned}$$

Note that this projection, contrary to exact marginalization, is not linear, and hence in general:

$$\text{Collapse}(p(s, \mathbf{x})q(\mathbf{x})) \neq \text{Collapse}(p(s, \mathbf{x}))q(\mathbf{x}).$$

In even more compact notation, with $\delta_{s,m}$ the Kronecker delta function, we can write a CG potential as

$$\begin{aligned} p(s, \mathbf{x}) &= \exp[\mathbf{v}^\top f(s, \mathbf{x})], \text{ with} \\ f(s, \mathbf{x}) &\equiv [\delta_{s,m} \delta_{s,m} \mathbf{x}^\top \delta_{s,m} \text{vec}(\mathbf{x}\mathbf{x}^\top)^\top | m \in S]^\top \\ \mathbf{v} &\equiv [g_s \mathbf{h}_s^\top - \frac{1}{2} \text{vec}(K_s)^\top | s \in S]^\top \end{aligned} \quad (15)$$

the sufficient statistics, and the canonical parameters respectively. In this notation the moment parameters follow from the canonical parameters as

$$g(\mathbf{v}) = \langle f(s, \mathbf{x}) \rangle_{\exp[\mathbf{v}^\top f(s, \mathbf{x})]} \equiv \sum_s \int d\mathbf{x} f(s, \mathbf{x}) \exp[\mathbf{v}^\top f(s, \mathbf{x})]. \quad (16)$$

Appendix B. The M-step Updates

We define θ as the set of all parameters

$$\theta \equiv \{ \Pi_{i \rightarrow j}, \mathbf{m}_1, V_1, A_j, C_j, \mu_j, r_j^2 | (i, j) \in G \},$$

and G as the set of allowed regime transitions

$$G \equiv \{ (n, n), (n, p), (n, s), (p, p), (p, f) \},$$

with the shorthands n, p, s, f, for normal, pre-fault, stop, and fault regimes respectively.

For now we assume a flat prior on θ , i.e. we compute ML instead of MAP estimates.

In the M -step we maximize the expected complete data log-likelihood $\hat{\mathcal{L}}$ with respect to θ . The expected complete data log-likelihood is defined as:

$$\hat{\mathcal{L}}(\mathbf{y}_{1:T}, s_{T+1} | \theta) \equiv E_{p(s_{1:T}, \mathbf{x}_{1:T} | \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}})} [\log p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}, s_{1:T+1} | \theta)].$$

Using the conditional independencies implied by the model and the constraints in the regime prior and transitions we can rewrite it as:

$$\begin{aligned} \hat{\mathcal{L}}(\mathbf{y}_{1:T}, s_{T+1} | \theta) &= p(s_1 = n | \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}}) E_{p(\mathbf{x}_1 | s_1 = n, \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}})} \log \mathcal{N}(\mathbf{x}_1; \mathbf{m}_1, V_1) \\ &+ \sum_{(i,j) \in G} \sum_{t=2}^{T+1} p(s_t = j, s_{t-1} = i | \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}}) \log \Pi_{i \rightarrow j} \\ &+ \sum_{j \in \{n, p\}} \sum_{t=2}^T p(s_t = j | \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}}) \\ &\quad E_{p(\mathbf{x}_{t-1,t} | s_t = j, \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}})} \log \mathcal{N}(\mathbf{x}_t; A_j \mathbf{x}_{t-1}, Q_j) \\ &+ \sum_{j \in \{n, p\}} \sum_{t=1}^T p(s_t = j | \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}}) \\ &\quad E_{p(\mathbf{x}_t | s_t = j, \mathbf{y}_{1:T}, s_{T+1}, \theta_{\text{old}})} \log \mathcal{N}(\mathbf{y}_t; C_j \mathbf{x}_t + \mu_j, r_j I). \end{aligned}$$

Note that from the model assumptions $p(s_1 = n | \mathbf{y}_{1:T}, s_{T+1}, \boldsymbol{\theta}_{\text{old}}) = 1$.

The M -step updates for the parameters follow by adding Lagrange multipliers for the normalization constraints and setting partial derivatives to 0.

We use $\langle \cdot \rangle$ to denote *weighted* expectations, and $p_t(ij)$ as a shorthand for the relevant posterior e.g.

$$\begin{aligned} \langle f(\mathbf{x}_{t-1}, \mathbf{x}_t) \rangle_{p_t(ij)} &= p(s_{t-1} = i, s_t = j | \mathbf{y}_{1:T}, s_{T+1}, \boldsymbol{\theta}_{\text{old}}) \\ &\quad \times \int d\mathbf{x}_{t-1,t} f(\mathbf{x}_{t-1}, \mathbf{x}_t) p(\mathbf{x}_{t-1,t} | s_{t-1} = i, s_t = j, \mathbf{y}_{1:T}, s_{T+1}, \boldsymbol{\theta}_{\text{old}}). \end{aligned}$$

In this notation $\langle 1 \rangle_{p_t(ij)}$ simply gives a weighting factor. In the statistics above, and hence in the update equations below, we recognize forms similar to a regular LDS but now with a weighting term that would not be present in the non-switching case.

The updates for $\Pi_{i \rightarrow j}$ are weighted versions of the standard HMM updates. The prior is deterministic (all sequences start in the normal regime) and fixed.

The updates read:

$$\begin{aligned} A_j^{\text{new}} &= \left(\sum_{t=2}^T \langle \mathbf{x}_t \mathbf{x}_{t-1}^\top \rangle_{p_t(\cdot j)} \right) \left(\sum_{t=2}^T \langle \mathbf{x}_{t-1} \mathbf{x}_{t-1}^\top \rangle_{p_t(\cdot j)} \right)^{-1} \\ Q_j^{\text{new}} &= \frac{\left(\sum_{t=2}^T \langle \mathbf{x}_t \mathbf{x}_t^\top \rangle_{p_t(\cdot j)} - A_j^{\text{new}} \sum_{t=2}^T \langle \mathbf{x}_t \mathbf{x}_{t-1}^\top \rangle_{p_t(\cdot j)}^\top \right)}{\sum_{t=2}^T \langle 1 \rangle_{p_t(\cdot j)}} \\ m_1^{\text{new}} &= \langle \mathbf{x}_1 \rangle_{p_1(n)} \\ V_1^{\text{new}} &= \langle \mathbf{x}_1 \mathbf{x}_1^\top \rangle_{p_1(n)} - m_1^{\text{new}} (m_1^{\text{new}})^\top \\ \Pi_{i \rightarrow j}^{\text{new}} &\propto \sum_{t=2}^{T+1} \langle 1 \rangle_{p_t(ij)} \quad \forall (i,j) \in G. \end{aligned}$$

We compute the new output matrix C_j and the new mean μ_j jointly by adding μ_j as an extra column to C_j and adding an entry to the continuous state that is always 1. We define

$$\begin{aligned} P_{t,j} &\equiv \begin{bmatrix} \langle \mathbf{x}_t \mathbf{x}_t^\top \rangle & \langle \mathbf{x}_t \rangle \\ \langle \mathbf{x}_t \rangle^\top & \langle 1 \rangle \end{bmatrix} \\ \mathbf{m}_{t,j} &\equiv \begin{bmatrix} \langle \mathbf{x}_t \rangle \\ \langle 1 \rangle \end{bmatrix} \\ \tilde{C}_j^{\text{new}} &\equiv [C_j^{\text{new}} \quad \mu_j^{\text{new}}], \end{aligned}$$

with the weighted expectations $\langle \cdot \rangle$ over $p_t(j)$, to arrive at

$$\begin{aligned} \tilde{C}_j^{\text{new}} &= \left(\sum_{t=1}^T \mathbf{y}_t \mathbf{m}_{t,j}^\top \right) \left(\sum_{t=1}^T P_{t,j} \right)^{-1} \\ r_j^{2 \text{ new}} &= \frac{\left(\sum_{t=1}^T \mathbf{y}_t^\top \mathbf{y}_t \langle 1 \rangle_{p_t(j)} - \text{tr} \left[\left(\tilde{C}_j^{\text{new}} \right)^\top \left(\sum_{t=1}^T \mathbf{y}_t \mathbf{m}_{t,j}^\top \right) \right] \right)}{d \sum_{t=1}^T \langle 1 \rangle_{p_t(j)}}, \end{aligned}$$

where d is the dimensionality of the observations \mathbf{y}_t .

When $s_{T+1} = f$, or if it is not observed, posterior distributions such as $p(\mathbf{x}_{t-1,t} | s_{t-1} = i, s_t = j, \mathbf{y}_{1:T}, s_{T+1} = f, \boldsymbol{\theta}_{\text{old}})$ are mixture of Gaussians (the $s_{T+1} = s$ case results in a straightforward LDS variant). For the updates described above first and second moments of these mixtures are required. They can be computed analytically and simply boil down to the weighted sum of the means and second moments of the individual components. For example, $\langle \mathbf{x}_t \mathbf{x}_{t-1}^\top \rangle_{p_t(\cdot|\mathbf{n})}$, is based on a mixture with $T - t - 1$ components (if s_{T+1} is observed to be a fault), each corresponding to a possible end of the normal regime.

$$\begin{aligned} \langle \mathbf{x}_t \mathbf{x}_{t-1}^\top \rangle_{p_t(\cdot|\mathbf{n})} &= \sum_{\tau=t}^{T-1} \langle \mathbf{x}_\tau \mathbf{x}_{\tau-1}^\top \rangle_{p_\tau(\cdot|\mathbf{n}:\mathbf{n})} \\ &\equiv \sum_{\tau=t}^{T-1} p(s_{1:\tau} = \mathbf{n} | \mathbf{y}_{1:T}, s_{T+1} = f, \boldsymbol{\theta}_{\text{old}}) \\ &\quad \times \int d\mathbf{x}_{t-1,t} \mathbf{x}_t \mathbf{x}_{t-1}^\top p(\mathbf{x}_{t-1,t} | s_{1:\tau} = \mathbf{n}, s_{T+1} = f, \boldsymbol{\theta}_{\text{old}}). \end{aligned}$$

If the trainset consists of V sequences instead of one, in the above update steps all sums $\sum_{t=a}^b$ are replaced by $\sum_{v=1}^V \sum_{t=a}^b$. Only the update for \mathbf{m}_1 and V_1 change. The posterior over \mathbf{x}_1 is a mixture of Gaussians with one mixture component for every sequence. The required sufficient statistics follow again by a collapse.

Appendix C. Prior Distributions

In practice, if the underlying models for normal and pre-fault regimes are relatively ‘‘far apart’’, we expect that the model parameters can be inferred reliably. For example if the pre-fault regime has an entirely different offset in the observation model, the pre-fault subsequences lie in an entirely different region of sensor space, which makes it easy to distinguish between the two. However in many practical applications we expect the difference not to be so profound. In this Section we introduce sensible priors on the parameters such that a priori knowledge can be incorporated.

Our main concern is with priors on the regime transition probabilities. There are three free parameters in the transition probabilities model: $\Pi_{n \rightarrow n}$, $\Pi_{n \rightarrow p}$ and $\Pi_{p \rightarrow p}$ ($\Pi_{n \rightarrow s} \equiv 1 - (\Pi_{n \rightarrow n} + \Pi_{n \rightarrow p})$, and $\Pi_{p \rightarrow f} \equiv 1 - \Pi_{p \rightarrow p}$ by construction).

The conjugate prior for $\Pi_{p \rightarrow p}$ is

$$p(\Pi_{p \rightarrow p} | v_p, \lambda_p) \propto (\Pi_{p \rightarrow p})^{v_p \lambda_p} (1 - \Pi_{p \rightarrow p})^{v_p}.$$

The parameters v_p and λ_p have a natural interpretation as the number of sequences and the average number of $p \rightarrow p$ transitions in a hypothesized set of ‘‘pseudo observed’’ sequences.

A similar reasoning holds for the parameters $\Pi_{n \rightarrow n}$, $\Pi_{n \rightarrow s}$, and $\Pi_{n \rightarrow p}$. Suppose we observe $V_{ns} + V_{np}$ sequences with on average \bar{l}_n $n \rightarrow n$ transitions, and V_{ns} of these ended in a stop and V_{np} switched to pre-fault. The probability of observing this set S of sequences is

$$p(S | \Pi_{n \rightarrow n}, \Pi_{n \rightarrow s}, \Pi_{n \rightarrow p}) = (\Pi_{n \rightarrow n})^{(V_{ns} + V_{np}) \bar{l}_n} (\Pi_{n \rightarrow s})^{V_{ns}} (\Pi_{n \rightarrow p})^{V_{np}}.$$

The conjugate prior is

$$p(\Pi_{n \rightarrow n}, \Pi_{n \rightarrow s}, \Pi_{n \rightarrow p} | v_{ns}, v_{np}, \lambda_n) \propto (\Pi_{n \rightarrow n})^{(v_{ns} + v_{np}) \lambda_n} (\Pi_{n \rightarrow s})^{v_{ns}} (\Pi_{n \rightarrow p})^{v_{np}}.$$

MAP estimates can be computed by changing the M-step slightly. Instead of maximizing the likelihood, the EM algorithm now aims to maximize

$$p(\boldsymbol{\theta}|\mathbf{y}_{1:T}, s_{T+1}) \propto p(\mathbf{y}_{1:T}, s_{T+1}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{v}_{np}, \mathbf{v}_{ns}, \lambda_n, \mathbf{v}_p, \lambda_p).$$

The E-step stays the same, but the M-step updates are now found by maximizing

$$\widehat{MAP}(\mathbf{y}_{1:T}, s_{T+1}, \boldsymbol{\theta}) \equiv \hat{\mathcal{L}}(\mathbf{y}_{1:T}, s_{T+1}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{v}_{np}, \mathbf{v}_{ns}, \lambda_n, \mathbf{v}_p, \lambda_p).$$

The required changes in the M-step updates are minor and intuitive. Only the update step for transition probabilities changes and becomes

$$\Pi_{i \rightarrow j}^{\text{new}} \propto \sum_{t=2}^{T+1} \langle \mathbf{1} \rangle_{p_t(ij)} + \mathbf{v}_{ij} \quad \forall (i,j) \in G,$$

where

$$\begin{aligned} \mathbf{v}_{nn} &\equiv (\mathbf{v}_{np} + \mathbf{v}_{ns})\lambda_n \\ \mathbf{v}_{pp} &\equiv \mathbf{v}_p\lambda_p \\ \mathbf{v}_{ps} &\equiv \mathbf{v}_p. \end{aligned}$$

Appendix D. The Fixed Point Interpretation of Algorithm 2

In this section we show that fixed points of Algorithm 2 are stationary points of the generalized EP free energy (13), and that the algorithm can be interpreted as fixed point iteration in dual space. The proof and intuition are analogous to the result that fixed points of loopy belief propagation can be mapped to extrema of the Bethe free energy (Yedidia et al., 2005).

Theorem 1 *The collection of beliefs $\hat{p}_t(\mathbf{z}_{t-1,t})$ and $\hat{q}_t(\mathbf{z}_t)$ form fixed points of Algorithm 2 if and only if they are zero gradient points of \mathcal{F}_{GEP} under the appropriate constraints.*

Proof The properties of the fixed points of message passing follow from the description of Algorithm 2. We get the CG form (15) of messages $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ and their relationship with one and two slice marginals

$$\begin{aligned} \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) &\propto \boldsymbol{\alpha}_{i-1}(\mathbf{u}_{\text{over}(i-1)})\Psi^{(i)}(\mathbf{u}_{\text{outer}(i)})\boldsymbol{\beta}_i(\mathbf{u}_{\text{over}(i)}) \\ \tilde{q}_i(\mathbf{u}_{\text{over}(i)}) &\propto \boldsymbol{\alpha}_i(\mathbf{u}_{\text{over}(i)})\boldsymbol{\beta}_i(\mathbf{u}_{\text{over}(i)}) \end{aligned}$$

by construction, and weak consistency

$$\langle f(\mathbf{u}_{\text{over}(i)}) \rangle_{\tilde{p}_i} = \langle f(\mathbf{u}_{\text{over}(i)}) \rangle_{\tilde{q}_i} = \langle f(\mathbf{u}_{\text{over}(i)}) \rangle_{\tilde{p}_{i+1}}, \quad (17)$$

as a property of a fixed point.

To identify the nature of stationary points of \mathcal{F}_{GEP} we first construct the Lagrangian by adding Lagrange multipliers $\alpha_i(\mathbf{u}_{\text{over}(i)})$ and $\beta_i(\mathbf{u}_{\text{over}(i)})$ for the forward and backward consistency constraints and $\gamma_{\text{outer}(i)}$ and $\gamma_{\text{over}(i)}$ for the normalization constraints.

$$\begin{aligned}
 \mathcal{L}_{\text{GEP}}(\tilde{p}, \tilde{q}, \alpha, \beta, \gamma) &= \sum_{i=1}^N \sum_{\mathbf{u}_{\text{outer}(i)}} \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) \log \frac{\tilde{p}_i(\mathbf{u}_{\text{outer}(i)})}{\Psi^{(i)}(\mathbf{u}_{\text{outer}(i)})} \\
 &\quad - \sum_{i=1}^{N-1} \sum_{\mathbf{u}_{\text{over}(i)}} \tilde{q}_i(\mathbf{u}_{\text{over}(i)}) \log \tilde{q}_i(\mathbf{u}_{\text{over}(i)}) \\
 &\quad - \sum_{i=2}^N \alpha_{i-1}(\mathbf{u}_{\text{over}(i-1)})^\top \left[\sum_{\mathbf{u}_{\text{outer}(i)}} f(\mathbf{u}_{\text{over}(i-1)}) \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) - \sum_{\mathbf{u}_{\text{over}(i-1)}} f(\mathbf{u}_{\text{over}(i-1)}) \tilde{q}_{i-1}(\mathbf{u}_{\text{over}(i-1)}) \right] \\
 &\quad - \sum_{i=1}^{N-1} \beta_i(\mathbf{u}_{\text{over}(i)})^\top \left[\sum_{\mathbf{u}_{\text{outer}(i)}} f(\mathbf{u}_{\text{over}(i)}) \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) - \sum_{\mathbf{u}_{\text{over}(i)}} f(\mathbf{u}_{\text{over}(i)}) \tilde{q}_i(\mathbf{u}_{\text{over}(i)}) \right] \\
 &\quad - \sum_{i=1}^N \gamma_{\text{outer}(i)} \left[\sum_{\mathbf{u}_{\text{outer}(i)}} \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) - 1 \right] - \sum_{i=1}^{N-1} \gamma_{\text{over}(i)} \left[\sum_{\mathbf{u}_{\text{over}(i)}} \tilde{q}_i(\mathbf{u}_{\text{over}(i)}) - 1 \right].
 \end{aligned}$$

Note that $\alpha_i(\mathbf{u}_{\text{over}(i)})$ and $\beta_i(\mathbf{u}_{\text{over}(i)})$ (in boldface to distinguish them from messages and to emphasize that they are vectors) are vectors of canonical parameters as defined in Appendix A.

The stationarity conditions follow by setting the partial derivatives to 0. Taking derivatives w.r.t. $\tilde{p}_i(\mathbf{u}_{\text{outer}(i)})$ and $\tilde{q}_i(\mathbf{u}_{\text{over}(i)})$ gives

$$\begin{aligned}
 \frac{\partial \mathcal{L}_{\text{GEP}}}{\partial \tilde{p}_i(\mathbf{u}_{\text{outer}(i)})} &= \log \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) + 1 - \log \Psi^{(i)}(\mathbf{u}_{\text{outer}(i)}) \\
 &\quad - \alpha_{i-1}(\mathbf{u}_{\text{over}(i-1)})^\top f(\mathbf{u}_{\text{over}(i-1)}) - \beta_i(\mathbf{u}_{\text{over}(i)})^\top f(\mathbf{u}_{\text{over}(i)}) - \gamma_{\text{outer}(i)} \\
 \frac{\partial \mathcal{L}_{\text{GEP}}}{\partial \tilde{q}_i(\mathbf{u}_{\text{over}(i)})} &= -\log \tilde{q}_i(\mathbf{u}_{\text{over}(i)}) - 1 + \alpha_i(\mathbf{u}_{\text{over}(i)})^\top f(\mathbf{u}_{\text{over}(i)}) + \beta_i(\mathbf{u}_{\text{over}(i)})^\top f(\mathbf{u}_{\text{over}(i)}) - \gamma_{\text{over}(i)}.
 \end{aligned}$$

Setting above derivatives to 0 and filling in the solutions for $\gamma_{\text{outer}(i)}$ and $\gamma_{\text{over}(i)}$ (which implies the normalization of the potentials) results in

$$\begin{aligned}
 \tilde{p}_i(\mathbf{u}_{\text{outer}(i)}) &\propto e^{\alpha_{i-1}(\mathbf{u}_{\text{over}(i-1)})^\top f(\mathbf{u}_{\text{over}(i-1)})} \Psi^{(i)}(\mathbf{u}_{\text{outer}(i)}) e^{\beta_i(\mathbf{u}_{\text{over}(i)})^\top f(\mathbf{u}_{\text{over}(i)})} \\
 \tilde{q}_i(\mathbf{u}_{\text{over}(i)}) &\propto e^{\alpha_i(\mathbf{u}_{\text{over}(i)})^\top f(\mathbf{u}_{\text{over}(i)}) + \beta_i(\mathbf{u}_{\text{over}(i)})^\top f(\mathbf{u}_{\text{over}(i)})}.
 \end{aligned}$$

The conditions $\frac{\partial \mathcal{L}_{\text{GEP}}}{\partial \alpha_i(\mathbf{u}_{\text{over}(i)})} = 0$ and $\frac{\partial \mathcal{L}_{\text{GEP}}}{\partial \beta_i(\mathbf{u}_{\text{over}(i)})} = 0$ retrieve the forward-equals-backward constraints (17).

So if we identify α_i as the vector of the canonical parameters of the message α_i and β_i as the vector of the canonical parameters of the message β_i , we see that the conditions for stationarity of \mathcal{F}_{GEP} and fixed points of Algorithm 2 are the same. \blacksquare

As can be seen from the above proof, iteration of the forward-backward passes can be interpreted as fixed point iteration in terms of Lagrange multipliers.

References

- Y. Bar-Shalom and X.-R. Li. *Estimation and Tracking: Principles, Techniques, and Software*. Artech House, 1993.
- A. T. Cemgil, H. J. Kappen, and D. Barber. A generative model for music transcription. *Accepted to IEEE Transactions on Speech and Audio Processing*, 2004.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.
- P. Fearnhead. Exact and efficient Bayesian inference for multiple changepoint problems. Technical report, Dept. of Math. and Stat., Lancaster University, 2003.
- P. J. Harrison and C. F. Stevens. Bayesian forecasting. *Journal of the Royal Statistical Society Society B*, 38:205–247, 1976.
- T. Heskes and O. Zoeter. Generalized belief propagation for approximate inference in hybrid Bayesian networks. In C. Bishop and B. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2003.
- Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI 2002)*, San Francisco, CA, 2002. Morgan Kaufmann Publishers.
- T. Jaakkola. Tutorial on variational approximation methods. In *Advanced Mean Field Methods, Theory and Practice*. MIT Press, 2001.
- H. J. Kappen and W. Wierwille. Novel iteration schemes for the cluster variation method. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 415–422, Cambridge, MA, 2002. MIT Press.
- C.-J. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60:1–22, 1994.
- P. R. Krishnaiah and B. Q. Miao. Review about estimation of change points. In P. R. Krishnaiah and C. R. Rao, editors, *Handbook of Statistics*, volume 7. Elsevier, 1988.
- F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.
- Steffen L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108, 1992.
- T. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI 2001)*. Morgan Kaufmann Publishers, 2001.
- T. Minka and Y. Qi. Tree-structured approximations by expectation propagation. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

- J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kaufmann, 1988.
- A. Schillings, B. van Wezel, T. Mulder, and J. Duysens. Mechanically induced stumbling during human treadmill walking. *Journal of Neuroscience Methods*, 67:11–17, 1996.
- R. H. Shumway and D. S. Stoffer. Dynamic linear models with switching. *Journal of the American Statistical Association*, 86:763–769, 1991.
- M. Welling, Y. W. Teh, and T. Minka. Structured regions graphs: morphing ep into gbp. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI-2005)*, Corvallis, Oregon, 2005. AUAI Press.
- M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer, 2nd edition, 1997.
- J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, 1989.
- J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, July 2005.
- O. Zoeter and T. Heskes. Deterministic approximate inference techniques for conditionally Gaussian state space models. Submitted, 2005.

Asymptotics in Empirical Risk Minimization

Leila Mohammadi

EURANDOM

Post Office Box 513

5600 MB Eindhoven

The Netherlands

MOHAMMADI@EURANDOM.TUE.NL

Sara van de Geer

Seminar für Statistik

ETH-Zentrum, LEO D11, 8092 Zürich

Switzerland

GEER@STAT.MATH.ETHZ.CH

Editor: John Shawe-Taylor

Abstract

In this paper, we study a two-category classification problem. We indicate the categories by labels $Y = 1$ and $Y = -1$. We observe a covariate, or feature, $X \in \mathcal{X} \subset \mathbb{R}^d$. Consider a collection $\{h_a\}$ of classifiers indexed by a finite-dimensional parameter a , and the classifier h_{a^*} that minimizes the prediction error over this class. The parameter a^* is estimated by the empirical risk minimizer \hat{a}_n over the class, where the empirical risk is calculated on a training sample of size n . We apply the Kim Pollard Theorem to show that under certain differentiability assumptions, \hat{a}_n converges to a^* with rate $n^{-1/3}$, and also present the asymptotic distribution of the renormalized estimator.

For example, let V_0 denote the set of x on which, given $X = x$, the label $Y = 1$ is more likely (than the label $Y = -1$). If X is one-dimensional, the set V_0 is the union of disjoint intervals. The problem is then to estimate the thresholds of the intervals. We obtain the asymptotic distribution of the empirical risk minimizer when the classifiers have K thresholds, where K is fixed. We furthermore consider an extension to higher-dimensional X , assuming basically that V_0 has a smooth boundary in some given parametric class.

We also discuss various rates of convergence when the differentiability conditions are possibly violated. Here, we again restrict ourselves to one-dimensional X . We show that the rate is n^{-1} in certain cases, and then also obtain the asymptotic distribution for the empirical prediction error.

Keywords: asymptotic distribution, classification theory, estimation error, nonparametric models, threshold-based classifiers

1. Introduction

In the theory of classification, the problem is to predict the unknown nature of a feature. The topic plays a basic role in several fields, such as data mining, artificial intelligence and neural networks. In this paper we discuss the classification problem from a parametric-statistical point of view.

Let the training set $(X_1, Y_1), \dots, (X_n, Y_n)$ consist of n independent copies of the couple (X, Y) with distribution P , where $X \in \mathcal{X} \subset \mathbb{R}^d$ is called a feature and $Y \in \{-1, 1\}$ is the label of X . A classifier h is a function $h : \mathcal{X} \rightarrow \{-1, 1\}$, attaching the label $h(X)$ to the feature X . The error, or risk, of a classifier h is defined as $P(h(X) \neq Y)$. Following Vapnik (2000) and Vapnik (1998), we

consider the empirical counterpart of the risk which is the number of misclassified examples, i.e.,

$$P_n(h(X) \neq Y) := \frac{1}{n} \sum_{i=1}^n \mathbb{1}(h(X_i) \neq Y_i).$$

Here, and throughout, $\mathbb{1}(A)$ denotes the indicator function of a set A . We will study empirical risk minimization over a model class \mathcal{H} of classifiers h . We take \mathcal{H} to be parametric, in the sense that

$$\mathcal{H} = \{h_a : a \in \mathcal{A}\},$$

with \mathcal{A} a subset of finite-dimensional Euclidean space.

Let

$$F_0(x) := P(Y = 1|X = x) \tag{1}$$

be the conditional probability of the label $Y = 1$ if the feature X has value x . Given a new feature $x \in \mathcal{X}$, we want to guess whether the label is $Y = 1$ or $Y = -1$. A natural solution is to predict $Y = 1$ when the label $Y = 1$ is more likely than the label $Y = -1$ (Bayes rule). Thus the set

$$V_0 := \{x \in \mathcal{X} : F_0(x) > 1/2\}, \tag{2}$$

plays a key role in classification. Bayes classifier is

$$h_0 = 2\mathbb{1}\{V_0\} - 1.$$

The collection \mathcal{H} of classifiers is viewed as model class for h_0 . However, we will not require that $h_0 \in \mathcal{H}$. If $h_0 \notin \mathcal{H}$, the model is misspecified.

In the statistical theory of classification, rates of convergence of empirical classifiers have been studied by a number of researchers, see for example Lugosi and Vayatis (2004), Lugosi and Nobel (1999), Lugosi and Wegkamp (2004), Koltchinskii and Panchenko (2002), Boucheron et al. (2005), Koltchinskii (2003a), Koltchinskii (2003b), Mohammadi (2004) and Tsybakov and van de Geer (2005). These papers generally consider a high-dimensional model class and use regularization to tackle the curse of dimensionality. Rates of convergence for the regularized estimators are obtained, and also non-asymptotic bounds. In this paper, we consider a low-dimensional model class. This means that we place the subject in the context of classical parametric statistics. Under regularity assumptions, one can establish rates, as well as the asymptotic distributions. Indeed, our main aim is to show that one can apply certain statistical results to the classification problem with parametric model class. In practice, one may not be willing to assume a simple parametric model class, as the complexity of the problem is not known a priori. In this sense, our study is primarily a theoretical one.

In Section 2, we generalize the problem considered in Mohammadi and van de Geer (2003). It gives an application of the cube root asymptotics derived by Kim and Pollard (1990). We briefly explain the main idea of the Kim Pollard Theorem. Its exact conditions are given in Section 4. We study in Subsection 2.1 the case where \mathcal{X} is one-dimensional. The set $V_0 \subset \mathbb{R}$ is then a union of disjoint intervals, and our aim is to estimate the boundaries of the intervals. These boundaries will be called thresholds. The situation that V_0 is the union of intervals has also been considered in Breiman et al. (1984). They explain how to use the training set to split the feature space \mathcal{X} and construct trees. See also Kearns et al. (1997) for a comparison of various algorithms in this case.

A simple case, with just one threshold, has been presented in Mohammadi and van de Geer (2003). We will establish the asymptotic behavior of estimators of the thresholds, using the set of classifiers with K thresholds as model class. Here K is fixed, and not bigger than, but not necessarily equal to, the number of thresholds of Bayes classifier. We moreover assume that F_0 is differentiable. In Subsection 2.2, we extend the situation to higher-dimensional feature space, $\mathcal{X} := \mathbb{R}^d$, $d \geq 1$. The problem there is related to assuming a single index model for the regression of Y on X , i.e.,

$$F_0(x) = \eta_0(x^T a^*),$$

where a^* is an unknown vector parameter, and η_0 is an unknown (monotone) function. We let $X = (U, V)$, with $U \in \mathbb{R}^{d-1}$ and $V \in \mathbb{R}$ and minimize the empirical classification error over the classifiers

$$h_a(u, v) := 2\mathbb{1}\{k_a(u) \geq v\} - 1,$$

where a is an r -dimensional parameter and $k_a : \mathbb{R}^{d-1} \rightarrow \mathbb{R}$ is some given smooth function of a . Under differentiability conditions, this will again lead to cube root asymptotics.

In Section 3, we study various other rates, and also the asymptotic distribution in the case of a $(1/n)$ -rate. We consider here only one-dimensional \mathcal{X} . The Kim Pollard Theorem and the proofs of the results in Section 2 are given in Section 4.

We note here that we will mainly concentrate on the estimation of the parameter a^* that minimizes the prediction error over the class \mathcal{H} . One may argue that the most interesting and useful subject is perhaps not the convergence of the estimator \hat{a}_n to a^* , but rather the convergence of the prediction error of (the classifier $h_{\hat{a}_n}$ corresponding to) \hat{a}_n . We remark however that our approach to study the former is via the latter. For example, in Corollary 2 the asymptotic distribution of the prediction error follows as a corollary.

The conclusion is that by considering some assumptions on the distribution of the data, we can prove rates of convergence and asymptotic distributions. In computer learning theory, usually no or minimal distributional assumptions are made. The results of the present paper give more insight in the dependency of the asymptotic behavior on the underlying distribution.

We consider asymptotics as $n \rightarrow \infty$, regarding the sample $(X_1, Y_1), \dots, (X_n, Y_n)$ as the first n of an infinite sequence of i.i.d. copies of (X, Y) . The distribution of the infinite sequence $(X_1, Y_1), (X_2, Y_2), \dots$ is denoted by \mathbf{P} . The marginal distribution function of X is denoted by G . In case that the density of the distribution G of X with respect to Lebesgue measure exists, it is denoted by g . The Euclidean norm is denoted by $\|\cdot\|$.

2. Cube Root Asymptotics

We first examine in Subsection 2.1 the case where the feature space \mathcal{X} is the unit interval in \mathbb{R} so that Bayes rule is the union of some subintervals in $[0, 1]$. As model class, we take the union of a, possibly smaller, number of subintervals. Next, we consider in Subsection 2.2 the situation where $\mathcal{X} = \mathbb{R}^d$ with $d > 1$. Our model class is then the class of graphs of smooth parametric functions. In both situations, the class of classifiers \mathcal{H} is parametric, i.e. it is of the form

$$\mathcal{H} = \{h_a : a \in \mathcal{A}\},$$

with \mathcal{A} a subset of \mathbb{R}^r , where the dimension r is fixed (not depending on n).

Define the empirical risk

$$L_n(a) := P_n(h_a(X) \neq Y), \quad (3)$$

and the theoretical risk

$$L(a) := P(h_a(X) \neq Y). \quad (4)$$

Moreover, let

$$\hat{a}_n = \arg \min_{a \in \mathcal{A}} L_n(a)$$

be the empirical risk minimizer, and let

$$a^* = \arg \min_{a \in \mathcal{A}} L(a)$$

be its theoretical counterpart. We assume that a^* exists and is unique. We also assume that the estimator \hat{a}_n exists, but it need not be unique. In fact, in the situations that we consider, there will be many solutions for \hat{a}_n . Our results will hold for any choice of \hat{a}_n .

We will derive cube root asymptotics. Let us first sketch where the $n^{-1/3}$ -rate of convergence comes from. One may write down the equality

$$L(\hat{a}_n) - L(a^*) = -[\mathbf{v}_n(\hat{a}_n) - \mathbf{v}_n(a^*)]/\sqrt{n} + [L_n(\hat{a}_n) - L_n(a^*)], \quad (5)$$

with

$$\mathbf{v}_n(a) = \sqrt{n}[L_n(a) - L(a)], \quad a \in \mathcal{A},$$

being the empirical process indexed by \mathcal{A} . Since $L_n(\hat{a}_n) - L_n(a^*) \leq 0$, this equality implies

$$L(\hat{a}_n) - L(a^*) \leq -[\mathbf{v}_n(\hat{a}_n) - \mathbf{v}_n(a^*)]/\sqrt{n}. \quad (6)$$

Under regularity conditions $L(a) - L(a^*)$ behaves like the squared distance $\|a - a^*\|^2$. Moreover, again under regularity conditions, the right hand side of (6) behaves in probability like $\sigma(\hat{a}_n)/\sqrt{n}$, where $\sigma(a)$ is the standard deviation of $[\mathbf{v}_n(a) - \mathbf{v}_n(a^*)]$. Due to the fact that we are dealing with indicator functions, the standard deviation of $[\mathbf{v}_n(a) - \mathbf{v}_n(a^*)]$ behaves like the *square root* $\|a - a^*\|^{1/2}$ of the distance between a and a^* . Inserting this in (6) yields that $\|\hat{a}_n - a^*\|^2$ is bounded by a term behaving in probability like $\|\hat{a}_n - a^*\|^{1/2}/\sqrt{n}$. But this implies $\|\hat{a}_n - a^*\|$ is of order $n^{-1/3}$ in probability.

Let us continue with a rough sketch of the arguments used for establishing the asymptotic distribution. We may write

$$\hat{a}_n = \arg \min_a \left[n^{\frac{1}{6}} [\mathbf{v}_n(a) - \mathbf{v}_n(a^*)] + n^{\frac{2}{3}} [L(a) - L(a^*)] \right].$$

When we already have the $n^{-1/3}$ -rate, it is convenient to renormalize to

$$n^{\frac{1}{3}}(\hat{a}_n - a^*) = \arg \min_t \left[n^{\frac{1}{6}} [\mathbf{v}_n(a^* + n^{-\frac{1}{3}}t) - \mathbf{v}_n(a^*)] + n^{\frac{2}{3}} [L(a^* + n^{-\frac{1}{3}}t) - L(a^*)] \right].$$

Now, under differentiability assumptions,

$$n^{\frac{2}{3}} [L(a^* + n^{-\frac{1}{3}}t) - L(a^*)] \approx t^T \mathcal{V}t/2,$$

where \mathcal{V} is the matrix of second derivatives of L at a^* . Moreover, the process $\{n^{1/6}[\mathbf{v}_n(a^* + n^{-1/3}t) - \mathbf{v}_n(a^*)] : t \in \mathbb{R}^r\}$ converges in distribution to some zero mean Gaussian process, say W . We then apply the ‘‘Argmax’’ Theorem (‘‘Argmin’’ Theorem in our case), see e.g., van der Vaart and Wellner (1996). The result is that $n^{1/3}(\hat{a}_n - a^*)$ converges in distribution to the location of the minimum of $\{W(t) + t^T \mathcal{V}t/2 : t \in \mathbb{R}^r\}$.

Kim and Pollard (1990) make these rough arguments precise. See Section 4 for the exact conditions.

2.1 One-Dimensional Feature Space

With a one-dimensional feature space, $\mathcal{X} = [0, 1]$, Bayes rule is described by the number, say K_0 , and the locations, say $a^0 = (a_1^0, \dots, a_{K_0}^0)^T$, where $2F_0 - 1$ changes sign. We call the locations of the sign changes *thresholds*. With a sign change we mean that the function has strictly opposite sign in sufficiently small intervals to the left and right side of each threshold. The boundary points $a_0^0 = 0$ and $a_{K_0+1}^0 = 1$ are thus not considered as locations of a sign change.

Let $K \in \mathbb{N}$ and U_K be the parameter space

$$U_K := \{a = (a_1, \dots, a_K) \in [0, 1]^K : a_1 < \dots < a_K\}. \tag{7}$$

Let for $a \in U_K$

$$h_a(x) := \sum_{k=1}^{K+1} b_k \mathbb{1}\{a_{k-1} \leq x < a_k\},$$

where $a_0 = 0$, $a_{K+1} = 1$ and $b_1 = -1$, $b_{k+1} = -b_k$, $k = 2, \dots, K$. Let \mathcal{H} be the collection of classifiers

$$\mathcal{H} = \{h_a : a \in U_K\}. \tag{8}$$

Let

$$L(a) := P(h_a(X) \neq Y), \quad L_n(a) := P_n(h_a(X) \neq Y). \tag{9}$$

The empirical risk minimizer is

$$\hat{a}_n := \arg \min_{a \in U_K} L_n(a). \tag{10}$$

We emphasize that we take the number of thresholds K in our model class fixed. Ideally, one would like to choose K equal to K_0 , but the latter may be unknown. Kearns et al. (1997), investigate an algorithm which calculates \hat{a}_n for all values of K , and a comparison of various regularization algorithms for estimating K_0 . With a consistent estimator \hat{K} in our model class, the asymptotics presented in this paper generally still go through. However, Kearns et al. (1997) and also later papers, e.g. Bartlett et al. (2002) show that the choice of K is very important in practice. Non-asymptotic bounds for a related problem are in Birgé (1987).

The following theorem states that \hat{a}_n converges to the minimizer a^* of $L(a)$ with rate $n^{-1/3}$ and also provides its asymptotic distribution after renormalization. We assume in this theorem that $K \leq K_0$. If $K = K_0$, one can show that when the minimizer a^* is unique, it is equal to a_0 , i.e., then h_{a^*} is Bayes classifier. The case $K < K_0$ is illustrated at the end of this subsection.

We use the notation $\mathbb{1}(u, v > 0)$ for $\mathbb{1}(u > 0)\mathbb{1}(v > 0)$, for scalars u and v . Likewise, we write $\mathbb{1}(u, v < 0)$ for $\mathbb{1}(u < 0)\mathbb{1}(v < 0)$.

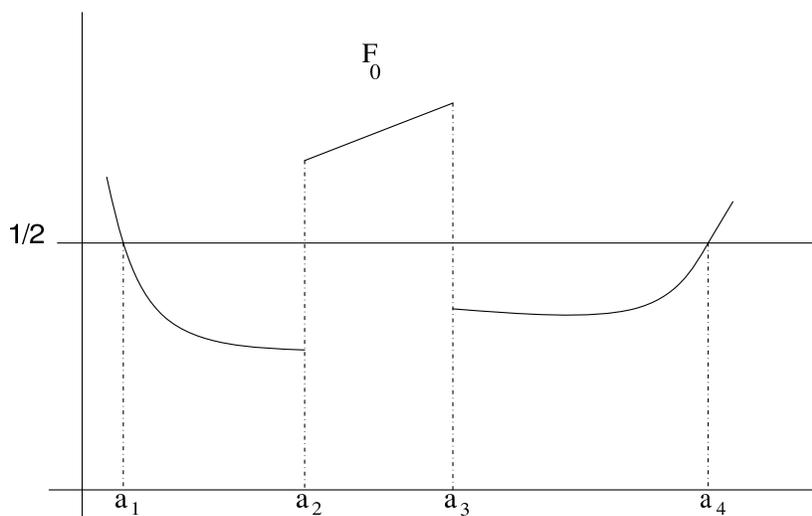


Figure 1: F_0 and the points at which $2F_0 - 1$ changes sign.

Theorem 1 Suppose $F_0(0) < 1/2$, that

$$a^* = (a_1^*, a_2^*, \dots, a_K^*) := \arg \min_{a \in U_K} L(a), \tag{11}$$

is the unique minimizer of $L(a)$, that a^* is in the interior of U_K , and that $L(a)$ is a continuous function of a . Suppose that F_0 has non-zero derivative f_0 in a neighborhood of a_k^* , $k = 1, \dots, K$. Let $g(a_k^*) > 0$, for all $k = 1, \dots, K$, where g , the density of G , is continuous in a neighborhood of a^* . Then the process

$$\{n^{2/3} [L_n(a^* + tn^{-1/3}) - L_n(a^*)] : t \in \mathbb{R}^K\}$$

(where we define $L_n(a) = 0$ for $a \notin U_K$), converges in distribution to a Gaussian process $\{Z(t) : t \in \mathbb{R}^K\}$ with continuous sample paths, and expected value $\mathbb{E}Z(t) = t^T \mathcal{V}t/2$, where

$$\mathcal{V} = \begin{bmatrix} 2f_0(a_1^*)g(a_1^*) & 0 & \dots & 0 \\ 0 & -2f_0(a_2^*)g(a_2^*) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & (-1)^{K-1}2f_0(a_K^*)g(a_K^*) \end{bmatrix},$$

and covariance kernel $H = [H(s, t)]$, where

$$H(s, t) = \sum_{k=1}^K g(a_k^*) [\min(s_k, t_k) \mathbb{1}(s_k, t_k > 0) - \max(s_k, t_k) \mathbb{1}(s_k, t_k < 0)].$$

Moreover,

$$n^{1/3}(\hat{a}_n - a^*) \rightarrow^{\mathcal{L}} \arg \min Z(t).$$

The proof can be found in Section 4, where it is also noted that the diagonal elements of the matrix \mathcal{V} are all positive.

Under the assumptions of Theorem 1

$$L(\hat{a}_n) - L(a^*) \approx (\hat{a}_n - a^*)^T \mathcal{V}(\hat{a}_n - a^*)/2$$

for large n . The theorem therefore also provides us the rate $n^{-2/3}$ for the convergence of the prediction error $L(\hat{a}_n)$ of the classifier $h_{\hat{a}_n}$, to the prediction error of h_{a^*} , and the asymptotic distribution of the prediction error $L(\hat{a}_n)$ after renormalization. We present this asymptotic distribution in a corollary.

Corollary 2 *Suppose the conditions of Theorem 1 are met. Then*

$$n^{\frac{2}{3}} [L(\hat{a}_n) - L(a^*)] \rightarrow^{\mathcal{L}} U^T \mathcal{V}U/2,$$

where $U = \arg \min_t Z(t)$, and Z is defined in Theorem 1.

Recall that one of the conditions in the above theorem is that L has a unique minimizer in the interior of U_K . This implies that K should not be larger than K_0 . Let us consider the situation $K = 1, K_0 = 2$ and discuss when there is a unique minimizer.

Suppose $K = 1$ and

$$F_0(x) \begin{cases} < 1/2 & x \notin [a_1^0, a_2^0], \\ > 1/2 & x \in (a_1^0, a_2^0), \end{cases} \quad (12)$$

where a_1^0 and a_2^0 are unknown and $0 < a_1^0 < a_2^0 < 1$. Note that

$$\begin{aligned} L(a) &= P(Y = 1, h_a(X) = -1) + P(Y = -1, h_a(X) = 1) \\ &= \int_0^a F_0 dG + \int_a^1 (1 - F_0) dG \\ &= \int_0^a (2F_0 - 1) dG + \int_0^1 (1 - F_0) dG. \end{aligned}$$

If $\int_{a_1^0}^1 (2F_0 - 1) dG > 0$, then $a^* = a_1^0$ is the unique minimizer of L . If $\int_{a_1^0}^1 (2F_0 - 1) dG < 0$, then L has a unique minimum at 1. The minimizer is not in the open interval $(0, 1)$, and Theorem 1 indeed fails. In this case, the convergence result is the same as Theorem 5 below (under its assumptions). If $\int_{a_1^0}^1 (2F_0 - 1) dG = 0$, then L has two minima at 1 and a_1^0 .

2.2 Higher-Dimensional Feature Space

In this subsection, $\mathcal{X} \subset \mathbb{R}^d$ with $d > 1$, and we write for $X \in \mathcal{X}$,

$$X = (U, V), \quad U \in \mathbb{R}^{d-1}, \quad V \in \mathbb{R}.$$

Consider given functions

$$k_a : \mathbb{R}^{d-1} \rightarrow \mathbb{R}, \quad a \in \mathcal{A},$$

and classifiers

$$h_a = 2\mathbb{1}\{C_a\} - 1, \quad a \in \mathcal{A},$$

where

$$C_a := \{(u, v) : v \leq k_a(u)\}, \quad a \in \mathcal{A}.$$

This kind of classifiers has been frequently considered and discussed in classification theory. We study the case where the parameter space is finite-dimensional, say $\mathcal{A} = \mathbb{R}^r$. A famous example is when k_a is linear in a , see for instance Hastie et al. (2001). Tsybakov and van de Geer (2005) consider this case for large r , depending on n . In contrast, we assume throughout that r is fixed.

Let again

$$a^* = \arg \min_a L(a),$$

be the minimizer of the theoretical risk $L(a)$, and

$$\hat{a}_n = \arg \min_a L_n(a)$$

be the empirical risk minimizer. We would like to know the asymptotic distribution of \hat{a}_n .

In this subsection, we suppose that the class $\{C_a : a \in \mathbb{R}^r\}$ is VC, i.e., that $\{k_a(u) : a \in \mathbb{R}^r\}$ is VC-subgraph. We also suppose that k_a is a regular function of the parameter $a \in \mathbb{R}^r$, i.e., the gradient

$$\frac{\partial}{\partial a} k_a(u) = k'_a(u) \tag{13}$$

of $k_a(u)$ exists for all u , and also its Hessian

$$\frac{\partial^2}{\partial a \partial a^T} k_a(u) = k''_a(u). \tag{14}$$

We will need to exchange the order of differentiation and integration of certain functions. To be able to do so, we require locally dominated integrability, which is defined as follows.

Definition 3 Let $\{f_a : a \in \mathcal{A}\}$, $\mathcal{A} \subset \mathbb{R}^r$, be a collection of functions on some measurable space (\mathcal{U}, μ) . It is called locally dominated integrable with respect to the measure μ and variable a if for each a there is a neighborhood I of a and a nonnegative μ -integrable function g_1 such that for all $u \in \mathcal{U}$ and $b \in I$,

$$|f_b(u)| \leq g_1(u).$$

The probability of misclassification using the classifier h_a is

$$\begin{aligned} L(a) &= P(h_a(X) \neq Y) = \int_{C_a} (1 - F_0) dG + \int_{C_a^c} F_0 dG \\ &= \int_{C_a} (1 - 2F_0) dG + P(Y = 1). \end{aligned}$$

Suppose that the density g of G , with respect to Lebesgue measure, exists. We use the notation

$$m(x) := (1 - 2F_0(x))g(x). \tag{15}$$

Assumption A: Assume existence of the derivatives (13) and (14) and also of

$$m'(u, v) := \frac{\partial}{\partial v} m(u, v).$$

Assume furthermore that the functions $m(u, k_a(u))k'_a(u)$ and $\frac{\partial}{\partial a^T} [m(u, k_a(u))k'_a(u)]$ are locally dominated integrable with respect to Lebesgue measure and variable a . Also, assume that the function $\int k'_a(u)g(u, k_a(u))du$ is uniformly bounded for a in a neighborhood of a^* , and that for each u , $m'(u, k_a(u))$ and $k''_a(u)$ are continuous in a neighborhood of a^* .

Write

$$\mathcal{V}_a := \frac{\partial^2}{\partial a \partial a^T} L(a).$$

Then

$$\mathcal{V}_a = \int \Sigma_a(u) m(u, k_a(u)) du, \quad (16)$$

where

$$\Sigma_a(u) = k'_a(u)k_a'^T(u) \frac{m'(u, k_a(u))}{m(u, k_a(u))} + k''_a(u). \quad (17)$$

In the following theorem, we show that $n^{\frac{1}{3}}(\hat{a}_n - a^*)$ converges to the location of the minimum of some Gaussian process.

Theorem 4 *Suppose that L has a unique minimum at a^* and that it is continuous at a^* . Assume that for all u , the density $g(u, v)$ is continuous as a function of v at $v = k_{a^*}(u)$. Let \mathcal{V}_a be continuous at a^* and $\mathcal{V} := \mathcal{V}_{a^*}$ be positive definite. Under Assumption A, we have*

$$n^{\frac{1}{3}}(\hat{a}_n - a^*) \rightarrow^{\mathcal{L}} \arg \min_{t \in \mathbb{R}^r} Z(t)$$

where $\{Z(t) : t \in \mathbb{R}^r\}$ is a Gaussian process with $\mathbb{E}Z(t) = t^T \mathcal{V} t / 2$, $t \in \mathbb{R}^r$, and with continuous sample paths and covariance structure

$$\text{Cov}(Z(t), Z(s)) = \int g(u, k_{a^*}(u)) \alpha^T(u, t, s) k'_{a^*}(u) du, \quad t, s \in \mathbb{R}^r,$$

with

$$\alpha(u, t, s) = \begin{cases} -s & t^T k'_{a^*}(u) \leq s^T k'_{a^*}(u) \leq 0 \\ -t & s^T k'_{a^*}(u) \leq t^T k'_{a^*}(u) \leq 0 \\ t & 0 \leq t^T k'_{a^*}(u) \leq s^T k'_{a^*}(u) \\ s & 0 \leq s^T k'_{a^*}(u) \leq t^T k'_{a^*}(u) \\ 0 & \text{o.w.} \end{cases} \quad (18)$$

The proof is given in Section 4.

As an example of Theorem 4, suppose $r = d$ and k_a is the linear function

$$k_a(u) := a_1 u_1 + \dots + a_{r-1} u_{r-1} + a_r.$$

It is interesting to compute the matrix \mathcal{V} (see (16) and (17)) in this case. Using our notations, we have

$$k'_a(u) = [u_1 \ u_2 \ \dots \ u_{r-1} \ 1]^T.$$

Let $f_0(u, v) := \frac{\partial}{\partial v} F_0(u, v)$ and $g'(u, v) := \frac{\partial}{\partial v} g(u, v)$ exist. Then by (15), we have

$$m'(u, v) = -2f_0(u, v)g(u, v) + (1 - 2F_0(u, v))g'(u, v)$$

and by (16) and (17)

$$\mathcal{V} = \left[\int u_i u_j (-2f_0(u, k_{a_0}(u))g(u, k_{a_0}(u)) + (1 - 2F_0(u, k_{a_0}(u)))g'(u, k_{a_0}(u))) du_1 \dots du_{r-1} \right],$$

where we define $u_r := 1$.

3. Other Rates of Convergence

In this section, we will investigate the rates that can occur if we do not assume the differentiability conditions needed for the Kim Pollard Theorem. We will restrict ourselves to the case of a one-dimensional feature space, with $\mathcal{X} = [0, 1]$.

We first assume $K = 1$, and that $2F_0 - 1$ has at most one sign change (i.e. $K_0 \leq 1$). Then, we briefly discuss what happens for general K_0 and K .

3.1 The Case of One Threshold and at Most One Sign Change

Let $K = 1$ and $K_0 \leq 1$. Now, either $2F_0 - 1$ changes sign at $a^* \in (0, 1)$ or there are no sign changes in $(0, 1)$, i.e. $K_0 = 0$. In the first case, we assume $F_0(x) < 1/2$ near 0. In the latter case, we assume $F_0(x) < 1/2$ for all $x \in (0, 1)$, and let $a^* = 1$, or $F_0(x) > 1/2$ for all $x \in (0, 1)$ and let $a^* = 0$. One easily verifies that a^* is the minimizer of $L(a)$ over $a \in [0, 1]$. However, if F_0 is not differentiable at a^* , Theorem 1 can not be applied. In this section, we impose the *margin condition* of Tsybakov (2004) (see also Mammen and Tsybakov (1999)). It can also be found on papers concerned with estimation of density level sets, see Polonik (1995) and Tsybakov (1997). In our context, this margin assumption is Assumption B below. Throughout, a neighborhood of a^* is some set of the form $(a^* - \delta, a^* + \delta)$, $\delta > 0$, intersected with $[0, 1]$.

Assumption B: Let there exist $c > 0$ and $\varepsilon \geq 0$ such that

$$|1 - 2F_0(x)|g(x) \geq c|x - a^*|^\varepsilon, \tag{19}$$

for all x in a neighborhood of a^* .

In Section 2, we assumed differentiability of F_0 in a neighborhood of $a^* \in (0, 1)$, with positive derivative f_0 . This corresponds to the case $\varepsilon = 1$. We have $\varepsilon = 0$ if F_0 has a jump at a^* , and also if $a^* \in \{0, 1\}$. In general, Assumption B describes how well a^* is identified: large values of ε correspond to less identifiability.

Recall now equality (6):

$$L(\hat{a}_n) - L(a^*) \leq -[\mathbf{v}_n(\hat{a}_n) - \mathbf{v}_n(a^*)] / \sqrt{n}. \tag{20}$$

Let $\sigma(a)$ be the standard deviation of $[\mathbf{v}_n(a) - \mathbf{v}_n(a^*)]$. Let

$$\Psi(r) = \mathbb{E} \left(\sup_{a: \sigma(a) \leq r} |\mathbf{v}_n(a) - \mathbf{v}(a)| \right), \quad r > 0. \tag{21}$$

It will follow from the proof of Theorem 5 below, that $\psi(r) \sim r$. Moreover, the standard deviation $\sigma(a)$ behaves like $\|a - a^*\|^{1/2}$. Therefore, as we already stated in Section 2, the right hand side of (20) behaves in probability like $\|\hat{a}_n - a^*\|^{1/2}/\sqrt{n}$. From Assumption B, we see that the left hand side behaves like $\|\hat{a}_n - a^*\|^{1+\varepsilon}$. This leads to the rate $n^{-\frac{1+\varepsilon}{1+2\varepsilon}}$.

Theorem 5 Consider the class \mathcal{H} defined in (8), with $K = 1$ and $b_1 = -1$. Under Assumption B,

$$\|\hat{a}_n - a^*\| = O_{\mathbf{P}}(n^{-\frac{1}{1+2\varepsilon}}), \quad L(\hat{a}_n) - L(a^*) = O_{\mathbf{P}}(n^{-\frac{1+\varepsilon}{1+2\varepsilon}}).$$

Proof We use the inequality (20):

$$L(\hat{a}_n) - L(a^*) \leq -[v_n(\hat{a}_n) - v_n(a^*)]/\sqrt{n}, \quad (22)$$

with $v_n(a) := \sqrt{n}[L_n(a) - L(a)]$. By Assumption B, we have the lower bound

$$L(\hat{a}_n) - L(a^*) \geq c\|\hat{a}_n - a^*\|^{1+\varepsilon}$$

for the left hand side of (22).

To find an upper bound for the right hand side of (20), we apply Theorem 5.12 of van de Geer (2000). Define

$$\mathcal{G} := \{\phi : \phi(x, y) := \mathbf{1}(h_a(x) \neq y), \quad a \in [0, 1]\}$$

and for $\phi^*(x, y) = \mathbf{1}(h_{a^*}(x) \neq y)$ and $\delta > 0$,

$$\mathcal{G}(\delta) := \{\phi - \phi^* : \phi \in \mathcal{G}, \|a - a^*\| \leq \delta^2\}.$$

Let $\{H_B(u, \mathcal{G}_1(\delta), P), u > 0\}$ be the entropy with bracketing, for the metric induced by the $L_2(P)$ norm, of the class $\mathcal{G}(\delta)$. It is easy to see that for some constant c_1 , and for all $\delta > 0$,

$$H_B(u, \mathcal{G}_1(\delta), P) \leq 2 \log \frac{c_1 \delta}{u}, \quad \forall u \in (0, \delta).$$

Set $\delta_n = n^{-1/2}$. We may select T, C, C_0 and C_1 such that for $a := C_1 T^2 \delta_n$ and $R := T \delta_n$, the conditions of Theorem 5.11 of van de Geer (2000) hold. This theorem then gives that for large T and large n ,

$$\mathbf{P} \left(\sup_{\|a - a^*\| \leq \delta_n^2} |v_n(a) - v_n(a^*)| \geq C_1 T^2 \delta_n \right) \leq C \exp(-T).$$

Now, by the peeling device, see for example van de Geer (2000), we can show that

$$\lim_{T \rightarrow \infty} \limsup_{n \rightarrow \infty} \mathbf{P} \left(\sup_{\sqrt{\|a - a^*\|} > \delta_n} \frac{|v_n(a) - v_n(a^*)|}{\sqrt{\|a - a^*\|}} \geq T \right) = 0.$$

So,

$$\frac{|v_n(\hat{a}_n) - v_n(a^*)|}{\sqrt{\|\hat{a}_n - a^*\|} \vee \delta_n} = O_{\mathbf{P}}(1). \quad (23)$$

Combining this with (22) and Assumption B yields

$$c\|\hat{a}_n - a^*\|^{1+\varepsilon} \leq (\sqrt{\|\hat{a}_n - a^*\|} + \delta_n) O_{\mathbf{P}}(1) / \sqrt{n}$$

or $\|\hat{a}_n - a^*\| = O_{\mathbf{P}}(n^{-1/(1+2\varepsilon)})$. Using (23) and (22), we can calculate $L(\hat{a}_n) - L(a^*) = O_{\mathbf{P}}(n^{-\frac{1+\varepsilon}{1+2\varepsilon}})$. ■

Theorem 5 can be refined to a non-asymptotic bound, for example in the following way. Let $\bar{\psi}$ be the smallest concave majorant of ψ defined in (21), and let $w(\cdot)$ be the smallest concave upper-bound of

$$r \mapsto \sup_{L(a) - L(a^*) \leq r^2} \sigma(a).$$

(In our situation, $w(r) \sim r^{\frac{1}{1+\varepsilon}}$.) Let r_* be the positive solution of

$$r^2 = \bar{\psi}(w(r)) / \sqrt{n}.$$

Then, from Massart (2003), Koltchinskii (2003b), or Bartlett et al. (2004), we obtain that

$$\mathbf{P}\left(L(\hat{a}_n) - L(a^*) > r_*^2 + \frac{w(r_*)}{r_*^2} \frac{2x}{n}\right) \leq e^{-x}, \quad x > 0.$$

When F_0 has a jump at a^* , we have the case $\varepsilon = 0$. Under the conditions of Theorem 5 with $\varepsilon = 0$, we derive the asymptotic distribution of the renormalized empirical risk, locally in a neighborhood of order $1/n$ of a^* , the local empirical risk. The rescaled estimator $n(\hat{a}_n - a^*)$ remains bounded in probability. However, since the local empirical risk has a limit law which has no unique minimum, $n(\hat{a}_n - a^*)$ generally does not converge in distribution. Similar results can be derived when a^* is one of the boundary points 0 or 1. For simplicity we only consider the right hand side limit. We assume that F_0 and g are right continuous.

In Theorem 6 below, convergence in distribution is to be understood in the sense given e.g. in Barbour et al. (1992).

Theorem 6 Consider the class \mathcal{H} defined in (8), with $K = 1$ and $b_1 = -1$. Assume that $a^* \in (0, 1)$, $1/2 < F_0(a^*) < 1$, g and F_0 are right continuous at a^* and $g(a^*) > 0$. Let

$$\lambda_1 := F_0(a^*)g(a^*), \quad \lambda_2 := (1 - F_0(a^*))g(a^*).$$

Let $Z_n(t) = n[L_n(a^* + t/n) - L_n(a^*)]$, $t > 0$. The process Z_n converges in distribution to $Z_1 - Z_2$, where Z_i is a Poisson process with intensity λ_i , $i = 1, 2$, and $Z_1(t)$ and $Z_2(s)$ are independent for all $s, t > 0$.

Proof We have for $t > 0$

$$Z_n(t) = \sum_{Y_i=1} \mathbb{1}(a^* \leq X_i < a^* + t/n) - \sum_{Y_i=-1} \mathbb{1}(a^* \leq X_i < a^* + t/n).$$

Define

$$I_n(t) := \sum_{Y_i=1} \mathbb{1}(a^* \leq X_i < a^* + t/n), \quad J_n(t) := \sum_{Y_i=-1} \mathbb{1}(a^* \leq X_i < a^* + t/n). \quad (24)$$

The random variable $I_n(t)$ has a binomial distribution with parameters n and p_1 , where

$$p_1 := P(Y = 1, a^* \leq X < a^* + t/n) = \int_{a^*}^{a^* + t/n} F_0 dG. \quad (25)$$

For large n , p_1 is close to $\lambda_1 t/n$. Similarly, for large n , $J_n(t)$ has binomial distribution with parameters n and $p_2 := \lambda_2 t/n$. We know that $B(n, \lambda t/n)$, for large n and small t , is approximately $\text{Poisson}(\lambda t)$, i.e. the total variation distance between the two distributions goes to zero as $n \rightarrow \infty$.

Note that for every $0 < t_1 < t_2 < 1$,

$$nP(Y = 1, a^* + t_1/n \leq X \leq a^* + t_2/n) = n \int_{a^* + t_1/n}^{a^* + t_2/n} F_0 dG \rightarrow \lambda_1(t_2 - t_1)$$

and

$$nP(Y = -1, a^* + t_1/n \leq X \leq a^* + t_2/n) = n \int_{a^* + t_1/n}^{a^* + t_2/n} (1 - F_0) dG \rightarrow \lambda_2(t_2 - t_1)$$

as $n \rightarrow \infty$. Now by Theorem 5.2.4, Remark 4 and Proposition A2.12 of Embrechts et al. (1997), we conclude that the whole process I_n (J_n) converges weakly to a Poisson process with intensity λ_1 (λ_2). (See also Barbour et al. (1992).) With the method of moment generating functions we can prove that the processes I_n and J_n are asymptotically independent, i.e., for any $t_1, \dots, t_m, s_1, \dots, s_k$,

$$E(\exp(r_1 I_n(t_1) + \dots + r_m I_n(t_m) + l_1 J_n(s_1) + \dots + l_k J_n(s_k)))$$

converges to

$$E(\exp(r_1 Z_1(t_1) + \dots + r_m Z_1(t_m))) E(\exp(l_1 Z_2(s_1) + \dots + l_k Z_2(s_k))).$$

Thus, $I_n - J_n$ converges weakly to the difference of two independent Poisson processes with intensities λ_1 and λ_2 . ■

3.2 Extension to Several Thresholds and Sign Changes

Recall that K_0 is the number of sign changes of $2F_0 - 1$, and that K is the number of thresholds in the model class \mathcal{H} defined in (8). Below, whenever we mention the rate $n^{-1/3}$ or n^{-1} , we mean the rate can be obtained under some conditions on F_0 and g (see Theorem 1 (where $\varepsilon = 1$), and Theorem 5 with $\varepsilon = 0$). Recall that a^0 denotes the K_0 -vector of the locations of the sign changes of $2F_0 - 1$.

1. Let $K \leq K_0$ and a^* is an interior point of U_K . In this case, \hat{a}_n converges to a^* . The rate is $n^{-1/3}$.

2. Let $K = K_0 + 1$. Then, K_0 of the elements of \hat{a}_n converge to a^0 , and either $\hat{a}_{1,n}$ converges to 0 or $\hat{a}_{K,n}$ converges to 1. The rate of convergence to the interior points is $n^{-1/3}$ and the rate of convergence to the boundary point is n^{-1} .

3. Let $K > K_0 + 1$. In this case, K_0 of the elements of \hat{a}_n converge to a^0 with rate $n^{-1/3}$. If $K - K_0$ is odd, one element of \hat{a}_n converges to one of the boundary points 0 or 1.

4. Proof of Theorem 1 and Theorem 4

We start out with presenting the Kim Pollard Theorem (Kim and Pollard (1990)) in a general context. Let ξ_1, ξ_2, \dots be a sequence of independent copies of a random variable ξ , with values in some space

\mathcal{S} . Let $\phi(\cdot, a) : \mathcal{S} \rightarrow \mathbb{R}$ be a collection of functions indexed by a parameter $a \in \mathcal{A} \subset \mathbb{R}^r$. Define $L_n(a) = \sum_{i=1}^n \phi(\xi_i, a)/n$ and $L(a) = E\phi(\xi, a)$. Moreover, let

$$v_n(a) = \sqrt{n}[L_n(a) - L(a)], \quad a \in \mathcal{A}.$$

Define

$$\mathcal{G}_R := \{\phi(\cdot, a) : |a_k - a_k^*| \leq R, k = 1, \dots, r\}, \quad R > 0. \tag{26}$$

The envelope G_R of this class is defined as

$$G_R(\cdot) = \sup_{\phi \in \mathcal{G}_R} |\phi(\cdot)|.$$

Theorem 1.1 in Kim and Pollard (1990) requires uniform manageability of a class of functions. The definition of uniform manageability can be found in Pollard (1989) and Pollard (1990). If \mathcal{G} is VC-subgraph, then a sufficient condition for the class \mathcal{G}_R to be uniformly manageable is that its envelope function G_R is uniformly square integrable for R near zero.

Theorem 7 (Kim and Pollard (1990)) *Let $\{\hat{a}_n\}$ be a sequence of estimators for which*

- (i) $L_n(\hat{a}_n) \leq \inf_{a \in \mathcal{A}} L_n(a) + o_P(n^{-2/3})$,
- (ii) \hat{a}_n converges in probability to the unique a^* that minimizes $L(a)$,
- (iii) a^* is an interior point of \mathcal{A} .

Let $\phi(\cdot, a^*) = 0$ and suppose

- (iv) $L(a)$ is twice differentiable with positive definite second derivative matrix \mathcal{V} at a^* ,
- (v) $H(s, t) = \lim_{\tau \rightarrow \infty} \tau E\phi(\xi, a^* + s/\tau)\phi(\xi, a^* + t/\tau)$ exists for each s, t in \mathbb{R}^d and

$$\lim_{\tau \rightarrow \infty} \tau E\phi(\xi, a^* + s/\tau)^2 \mathbb{1}\{|\phi(\xi, a^* + s/\tau)| > \eta\tau\} = 0$$

for each $\eta > 0$ and s in \mathbb{R}^r ,

- (vi) $E|\phi(\xi, a) - \phi(\xi, b)| = O(\|a - b\|)$ near a^* ,
- (vii) the classes \mathcal{G}_R in (26), for R near zero, are uniformly manageable for the envelopes G_R and satisfy $E(G_R^2) = O(R)$ as $R \rightarrow 0$, and for each $\eta > 0$ there is a constant C such that $E(G_R^2 \mathbb{1}\{G_R > C\}) < \eta R$ for R near zero.

Then the process $\{n^{2/3}[L_n(a^* + tn^{-1/3}) - L_n(a^*)] : t \in \mathbb{R}^r\}$, (where we take $L_n(a) = 0$ if $a \notin \mathcal{A}$), converges in distribution to a Gaussian process $\{Z(t) : t \in \mathbb{R}^r\}$ with continuous sample paths, expected value $\mathbb{E}Z(t) = t^T \mathcal{V}t/2$ and covariance kernel H . If Z has non-degenerate increments, then $n^{1/3}(\hat{a}_n - a^*)$ converges in distribution to the (almost surely unique) random vector that minimizes $\{Z(t) : t \in \mathbb{R}^r\}$.

Proof of Theorem 1 We apply the Kim Pollard Theorem to the function

$$\phi(x, y, a) := \mathbb{1}(h_a(x) \neq y) - \mathbb{1}(h_{a^*}(x) \neq y),$$

Condition (i) is met by the definition of \hat{a}_n . To check Condition (ii), we note that, because $\{\phi(\cdot, a) : a \in U_K\}$ is a uniformly bounded VC-subgraph class, we have the uniform law of large numbers

$$\sup_{a \in U_K} |L_n(a) - L(a)| \rightarrow 0, \text{ a.s..}$$

Since we assume that $a^* \in U_K$ is unique and L is continuous., this implies

$$\hat{a}_n \rightarrow a^*, \text{ a.s..}$$

Condition (iii) is satisfied by assumption.

To check Condition (iv), for odd i , we have

$$\frac{\partial}{\partial a_i} P(h_a(X) \neq Y) = (2F_0(a_i) - 1)g(a_i)$$

so

$$\begin{aligned} \frac{\partial^2}{\partial a_i^2} P(h_a(X) \neq Y) \Big|_{a_i=a_i^*} &= \left([2f_0(a_i)g(a_i) + (2F_0(a_i) - 1)g'(a_i)] \right) \Big|_{a_i=a_i^*} \\ &= -2f_0(a_i^*)g(a_i^*). \end{aligned}$$

For even i , these terms are symmetric. Thus (iv) is satisfied with

$$\mathcal{V} := \begin{bmatrix} 2f_0(a_1^*)g(a_1^*) & 0 & \dots & 0 \\ 0 & -2f_0(a_2^*)g(a_2^*) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & (-1)^{K-1}2f_0(a_K^*)g(a_K^*) \end{bmatrix}.$$

Now, a^* minimizes $L(a)$ for a in the interior of U_K , so $2F_0 - 1$ changes sign from negative to positive at a_k^* for odd k , and it changes sign from positive to negative at a_k^* for even k . Hence $f_0(a_k^*) > 0$ for odd k and $f_0(a_k^*) < 0$ for even k and therefore \mathcal{V} is positive definite.

Next, we study the existence of the covariance kernel H , required in Condition (v). Consider $t, s \in \mathbb{R}$ and large $\tau > 0$ so that $a^* + t/\tau, a^* + s/\tau \in U_K$. First we note that the product of the brackets is the same for $Y = 1$ and for $Y = -1$. For $a_1 < a_2, b_1 < b_2, a_1^* < a_2^*$, we have

$$\begin{aligned} &\left[\mathbf{1}(a_1^* \leq x < a_2^*) - \mathbf{1}(a_1 \leq x < a_2) \right] \left[\mathbf{1}(a_1^* \leq x < a_2^*) - \mathbf{1}(b_1 \leq x < b_2) \right] \\ &= \left[\mathbf{1}(x \geq a_1) - \mathbf{1}(x \geq a_2) - \mathbf{1}(x \geq a_1^*) + \mathbf{1}(x \geq a_2^*) \right] \\ &\quad \times \left[\mathbf{1}(x \geq b_1) - \mathbf{1}(x \geq b_2) - \mathbf{1}(x \geq a_1^*) + \mathbf{1}(x \geq a_2^*) \right] \\ &= A(x) - B(x) - C(x) + D(x), \end{aligned}$$

where

$$\begin{aligned} A(x) &:= (\mathbf{1}(x \geq a_1) - \mathbf{1}(x \geq a_1^*))(\mathbf{1}(x \geq b_1) - \mathbf{1}(x \geq a_1^*)) \\ &= \mathbf{1}[\min(a_1, a_1^*), \max(a_1, a_1^*)] \mathbf{1}[\min(b_1, a_1^*), \max(b_1, a_1^*)] \\ &= \mathbf{1}[a_1^*, \min(a_1, b_1)] \mathbf{1}(a_1^* < \min(a_1, b_1)) + \mathbf{1}[\max(a_1, b_1), a_1^*] \mathbf{1}(a_1^* > \max(a_1, b_1)), \\ D(x) &:= (\mathbf{1}(x \geq a_2) - \mathbf{1}(x \geq a_2^*))(\mathbf{1}(x \geq b_2) - \mathbf{1}(x \geq a_2^*)) \\ &= \mathbf{1}[\min(a_2, a_2^*), \max(a_2, a_2^*)] \mathbf{1}[\min(b_2, a_2^*), \max(b_2, a_2^*)] \\ &= \mathbf{1}[a_2^*, \min(a_2, b_2)] \mathbf{1}(a_2^* < \min(a_2, b_2)) + \mathbf{1}[\max(a_2, b_2), a_2^*] \mathbf{1}(a_2^* > \max(a_2, b_2)), \end{aligned}$$

$$B(x) := (\mathbb{1}(x \geq a_1) - \mathbb{1}(x \geq a_1^*))(\mathbb{1}(x \geq b_2) - \mathbb{1}(x \geq a_2^*)),$$

and

$$C(x) := (\mathbb{1}(x \geq a_2) - \mathbb{1}(x \geq a_2^*))(\mathbb{1}(x \geq b_1) - \mathbb{1}(x \geq a_1^*)).$$

Assume that $a_1 = a_1^* + s_1/\tau, a_2 = a_2^* + s_2/\tau, b_1 = a_1^* + t_1/\tau, b_2 = a_2^* + t_2/\tau$. When τ tends to infinity, we have $\int BdG = \int CdG = 0$. Moreover,

$$\begin{aligned} & \int (A+D)dG \\ &= \left[\mathbb{1}(0 < s_1, t_1) \int_{a_1^*}^{a_1^* + \min(s_1, t_1)/\tau} dG + \mathbb{1}(0 > s_1, t_1) \int_{a_1^* + \max(s_1, t_1)/\tau}^{a_1^*} dG \right. \\ & \quad \left. + \mathbb{1}(0 < s_2, t_2) \int_{a_2^*}^{a_2^* + \min(s_2, t_2)/\tau} dG + \mathbb{1}(0 > s_2, t_2) \int_{a_2^* + \max(s_2, t_2)/\tau}^{a_2^*} dG \right] \\ &= \min(s_1, t_1)g(a_1^*)\mathbb{1}(0 < s_1, t_1) - \max(s_1, t_1)g(a_1^*)\mathbb{1}(0 > s_1, t_1) \\ & \quad + \min(s_2, t_2)g(a_2^*)\mathbb{1}(0 < s_2, t_2) - \max(s_2, t_2)g(a_2^*)\mathbb{1}(0 > s_2, t_2). \end{aligned} \tag{27}$$

Let m be the integer part of $(K + 1)/2$. Now, we obtain

$$\begin{aligned} & E\phi(X, Y, a^* + s/\tau)\phi(X, Y, a^* + t/\tau) \\ &= E \left[\mathbb{1}(X \in \cup_{i=1}^m [a_{2i-1}^* + s_{2i-1}/\tau, a_{2i}^* + s_{2i}/\tau]) - \mathbb{1}(X \in \cup_{i=1}^m [a_{2i-1}^*, a_{2i}^*]) \right] \\ & \quad \times \left[\mathbb{1}(X \in \cup_{i=1}^m [a_{2i-1}^* + t_{2i-1}/\tau, a_{2i}^* + t_{2i}/\tau]) - \mathbb{1}(X \in \cup_{i=1}^m [a_{2i-1}^*, a_{2i}^*]) \right] \\ &= \sum_{k=1}^K E \left[\mathbb{1}(X \in [a_k^*, a_k^* + \min(s_k, t_k)])\mathbb{1}(0 < s_k, t_k) \right. \\ & \quad \left. - \mathbb{1}(X \in [a_k^* + \max(s_k, t_k), a_k^*])\mathbb{1}(0 > s_k, t_k) \right] \end{aligned} \tag{28}$$

(for large τ). Finally, by (27) and (28), the limit of $\tau E\phi(X, Y, a^* + s/\tau)\phi(X, Y, a^* + t/\tau)$ as $\tau \rightarrow \infty$ becomes

$$\begin{aligned} H(s, t) &= \sum_{k=1}^K \left[\min(s_k, t_k)g(a_k^*)\mathbb{1}(0 < s_k, t_k) \right. \\ & \quad \left. - \max(s_k, t_k)g(a_k^*)\mathbb{1}(0 > s_k, t_k) \right]. \end{aligned}$$

So, the first part of condition (v) is satisfied. As for the second part of condition (v), for any ε and $\tau > 1/\varepsilon$, and $t \in \mathbb{R}$, we have

$$E \left[\mathbb{1}^2(h_{a^* + t/\tau}(X) \neq Y)\mathbb{1}(\mathbb{1}(h_{a^* + t/\tau}(X) \neq Y) > \tau\varepsilon) \right] = 0.$$

To show that Condition (vi) is satisfied, we note that for any $a, b \in U_K$,

$$E \left[\left| \mathbb{1}(h_a(X) \neq Y) - \mathbb{1}(h_b(X) \neq Y) \right| \right] \leq \sum_{k=1}^K E \left[\mathbb{1}(X \in [\min(a_k, b_k), \max(a_k, b_k)]) \right]$$

$$\leq \sum_{k=1}^K |a_k - b_k| g(\xi_k)$$

for some $\xi_k \in [\min(a_k, b_k), \max(a_k, b_k)]$. Hence

$$E\left(|\mathbb{1}(h_a(X) \neq Y) - \mathbb{1}(h_b(X) \neq Y)|\right) = O(\|a - b\|),$$

for a and b near a^* .

Now we calculate an upper bound for the envelope function. Fix $(x, y) \in \mathcal{X} \times \{-1, 1\}$. To maximize the function $\phi(x, y, a) = |\mathbb{1}(h_a(x) \neq y) - \mathbb{1}(h_{a^*}(x) \neq y)|$, note that for $y = 1$, this function is increasing in a_k 's for even k and decreasing in a_k 's for odd k . To simplify, assume K is odd. Over \mathcal{G}_R , $\phi(x, y, a)$ is maximized when

$$a_1 = a_1^* - R, \quad a_2 = a_2^* + R, \quad a_3 = a_3^* - R, \quad \dots, \quad a_K = a_K^* - R. \tag{29}$$

For $y = -1$, it is maximized when

$$a_1 = a_1^* + R, \quad a_2 = a_2^* - R, \quad a_3 = a_3^* + R, \quad \dots, \quad a_K = a_K^* + R. \tag{30}$$

Similarly, $\mathbb{1}(h_{a^*}(x) \neq y) - \mathbb{1}(h_a(x) \neq y)$ is maximized for $y = 1$, in case (30) and for $y = -1$, it is maximized in case (29). So, the maximum of $|\phi(x, y, a)|$ is the maximum of

$$\mathbb{1}\left(x \in [a_1^* - R, a_1^*] \cup [a_2^*, a_2^* + R] \cup \dots \cup [a_K^* - R, a_K^*]\right)$$

and

$$\mathbb{1}\left(x \in [a_1^*, a_1^* + R] \cup [a_2^* - R, a_2^*] \cup \dots \cup [a_K^*, a_K^* + R]\right).$$

So the envelope G_R of \mathcal{G}_R satisfies

$$G_R \leq G'_R$$

where

$$G'_R = \mathbb{1}\left(x \in \cup_{k=1}^K [a_k^* - R, a_k^* + R]\right).$$

Now, note that

$$E(G_R^2) \leq \sum_{k=1}^K P(a_k^* - R \leq X \leq a_k^* + R)$$

and

$$\frac{P(a_k^* - R \leq X \leq a_k^* + R)}{R} = \frac{2Rg(a'_k)}{R} < R^*, \quad \exists R^* < \infty,$$

for some $a'_k \in (a_k^* - R, a_k^* + R)$, when R is close to zero. We thus have $E(G_R^2) = O(R)$. Since G'_R is bounded by one, it is also easy to see that G'_R is uniformly square integrable for R close to zero. Finally, since \mathcal{G} is VC-subgraph, we conclude that \mathcal{G}_R is uniformly manageable for the envelope G_R . ■

Proof of Theorem 4

Checking the Conditions (i)-(vii) of the Kim Pollard Theorem is very similar to the proof of Theorem 1. We consider again $\phi(x, y, a) = P(h_a(X) \neq Y) - P(h_{a^*}(X) \neq Y)$. Condition (i) is clearly true. Because the class $\{C_a : a \in \mathbb{R}^r\}$ is VC and L is continuous at a^* , we know by the same argument as in the proof of Theorem 1 that $\hat{a}_n \rightarrow a^*$ almost surely. So, Condition (ii) is met. Condition (iii) is met because \mathbb{R}^r is open. The function L is twice differentiable with positive definite second derivative matrix \mathcal{V} at a^* . So, (iv) is satisfied. To show that (v) is satisfied, we consider the covariance structure of $\phi(X, Y, a)$. Now,

$$\text{Cov}(\phi(X, Y, a), \phi(X, Y, \tilde{a})) = I - II,$$

where

$$I := E[\phi(X, Y, a)\phi(X, Y, \tilde{a})]$$

and

$$II := [L(a) - L(a^*)][L(\tilde{a}) - L(a^*)] = O(\tau^{-4}),$$

for $\|a - a^*\| = O(1/\tau)$ and $\|\tilde{a} - a^*\| = O(1/\tau)$. As for I , write $C = C_a$, $\tilde{C} = C_{\tilde{a}}$, and $C_* = C_{a^*}$, then

$$\begin{aligned} I &= P(Y = 1, X \in C^c \cap \tilde{C}^c) - P(Y = 1, X \in C^c \cap C_0^c) \\ &\quad - P(Y = 1, X \in C_0^c \cap \tilde{C}^c) + P(Y = 1, X \in C_*^c) \\ &+ P(Y = -1, X \in C \cap \tilde{C}) - P(Y = -1, X \in C \cap C_*) \\ &\quad - P(Y = -1, X \in C_0 \cap \tilde{C}) + P(Y = -1, X \in C_*). \end{aligned}$$

It is easy to see that

$$\begin{aligned} I &= \int \left[\int_{v \geq k_a(u), v \geq k_{\tilde{a}}(u)} F_0(u, v) - \int_{v \geq k_a(u), v \geq k_{a^*}(u)} F_0(u, v) \right. \\ &\quad \left. - \int_{v \geq k_{a^*}(u), v \geq k_{\tilde{a}}(u)} F_0(u, v) + \int_{v \geq k_{a^*}(u)} F_0(u, v) \right. \\ &\quad \left. + \int_{v < k_a(u), v < k_{\tilde{a}}(u)} (1 - F_0(u, v)) - \int_{v < k_a(u), v < k_{a^*}(u)} (1 - F_0(u, v)) \right. \\ &\quad \left. - \int_{v < k_{a^*}(u), v < k_{\tilde{a}}(u)} (1 - F_0(u, v)) + \int_{v < k_{a^*}(u)} (1 - F_0(u, v)) \right] g(u, v) dudv. \\ &= \int_{k_a(u) \leq k_{\tilde{a}}(u) \leq k_{a^*}(u)} \int_{k_{\tilde{a}}(u)}^{k_{a_0}(u)} g(u, v) dv du + \int_{k_{\tilde{a}}(u) \leq k_a(u) \leq k_{a^*}(u)} \int_{k_a(u)}^{k_{a^*}(u)} g(u, v) dv du \\ &\quad + \int_{k_{a^*}(u) \leq k_a(u) \leq k_{\tilde{a}}(u)} \int_{k_{a^*}(u)}^{k_a(u)} g(u, v) dv du + \int_{k_{a^*}(u) \leq k_{\tilde{a}}(u) \leq k_a(u)} \int_{k_{a^*}(u)}^{k_{\tilde{a}}(u)} g(u, v) dv du. \end{aligned}$$

For each $s, t \in \mathbb{R}^r$, and for sequences $\{\bar{a}(\tau)\}$ and $\{\underline{a}(\tau)\}$ with

$$\lim_{\tau \rightarrow \infty} \bar{a}(\tau) = \lim_{\tau \rightarrow \infty} \underline{a}(\tau) = a^*,$$

we have

$$\lim_{\tau \rightarrow \infty} \tau \int_{k_{a^*+s/\tau}(u) \leq k_{a^*+t/\tau}(u) \leq k_{a^*}(u)} \int_{k_{a^*+t/\tau}(u)}^{k_{a^*}(u)} g(u, v) dv du$$

$$\begin{aligned}
&= \lim_{\tau \rightarrow \infty} \tau \int_{k_{a^*+s/\tau}(u) \leq k_{a^*+t/\tau}(u) \leq k_{a^*}(u)} \left(k_{a^*}(u) - k_{a^*+t/\tau}(u) \right) g(u, k_{\bar{a}(\tau)}(u)) du \\
&= \lim_{\tau \rightarrow \infty} \tau \int_{k_{a^*+s/\tau}(u) \leq k_{a^*+t/\tau}(u) \leq k_{a^*}(u)} (-t^T/\tau) k'_{\underline{a}(\tau)}(u) g(u, k_{\bar{a}(\tau)}(u)) du. \tag{31}
\end{aligned}$$

When $\tau \rightarrow \infty$, the conditions $k_{a_0+s/\tau}(u) \leq k_{a^*+t/\tau}(u)$ and $k_{a^*+t/\tau}(u) \leq k_{a^*}(u)$ becomes $(-s^T + t^T)k'_{a^*}(u) \geq 0$ and $-t^T k'_{a^*}(u) \geq 0$, respectively. So the limit in (31) becomes

$$- \int_{0 \geq t^T k'_{a^*}(u) \geq s^T k'_{a^*}(u)} t^T k'_{a^*}(u) g(u, k_{a^*}(u)) du.$$

Hence, have shown that

$$\begin{aligned}
&\lim_{\tau \rightarrow \infty} \tau \text{Cov}(\phi(X, Y, a^* + s/\tau), \phi(X, Y, a^* + t/\tau)) \\
&= \int \alpha^T(u, t, s) k'_{a^*}(u) g(u, k_{a^*}(u)) du,
\end{aligned}$$

where α is defined in (18). The second part of Condition (v) is true because the functions $\phi(\cdot, a)$ are bounded. We conclude that Condition (v) is satisfied.

Conditions (vi) and (vii) are verified in the same way as in the proof of Theorem 1. ■

References

- A. Barbour, L. Holst, and S. Janson. *Poisson Approximation*. Oxford Studies in Probability. Clarendon Press, Oxford, 1992.
- P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, 2002.
- P. L. Bartlett, S. Mendelson, and P. Philipps. Empirical risk minimization. *Proceedings of COLT 2004*, Springer Verlag, 2004.
- L. Birgé. Estimating a density under order restrictions. *Ann. Statist.*, 15(3):995–1012, 1987.
- S. Boucheron, G. Bousquet, and G. Lugosi. Theory of classification: some recent advances. *To appear in ESAIM Probability and statistics*, 2005.
- L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont, CA, 1984.
- P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling extremal events. For insurance and finance*, volume 33 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1997.
- T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data mining, Inference, and Prediction*. Springer Series in Statistics. Springer-Verlag, New York, 2001.
- M. Kearns, Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27:7–50, 1997.

- J. Kim and D. Pollard. Cube root asymptotics. *Ann. Statist.*, 18(1):191–219, 1990.
- V. Koltchinskii. Bounds on margin distributions in learning problems. *Ann. Inst. H. Poincaré Probab. Statist.*, 39(6):943–978, 2003a.
- V. Koltchinskii. Local rademacher complexities and oracle inequalities in risk minimization. *Preprint*, 2003b.
- V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Ann. Statist.*, 30(1):1–50, 2002.
- G. Lugosi and A. B. Nobel. Adaptive model selection using empirical complexities. *Ann. Statist.*, 27(6):1830–1864, 1999.
- G. Lugosi and N. Vayatis. On the bayes-risk consistency of regularized boosting methods. *Ann. Statist.*, 32, 2004.
- G. Lugosi and M. Wegkamp. Complexity regularization via localized random penalties. *Ann. Statist.*, 32(4):1679–1697, 2004.
- E. Mammen and A. B. Tsybakov. Smooth discriminant analysis. *Ann. Statist.*, 27(6):1808–1829, 1999.
- P. Massart. *Concentration inequalities and model selection*. Ecole d’Eté Probabilité de Saint Flour XXXIII. Springer Verlag, 2003.
- L. Mohammadi. *Estimation of thresholds in classification*. PhD thesis, University of Leiden, 2004.
- L. Mohammadi and S. A. van de Geer. On threshold-based classification rules. *Institute of Mathematical Statistics, Lecture Notes Monograph Series, Mathematical Statistics and Applications: Festschrift for Constance van Eeden*, 42:261–280, 2003.
- D. Pollard. Asymptotics via empirical processes. *Statist. Sci.*, 4(4):341–366, 1989. With comments and a rejoinder by the author.
- D. Pollard. *Empirical Processes: Theory and Applications*. NSF-CBMS Regional Conference Series in Probability and Statistics, 2. Institute of Mathematical Statistics, Hayward, CA, 1990.
- W. Polonik. Measuring mass concentrations and estimating density contour clusters-an excess mass approach. *Ann. Statist.*, 23(3):855–881, 1995.
- A. B. Tsybakov. On nonparametric estimation of density level sets. *Ann. Statist.*, 25(3):948–969, 1997.
- A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Ann. Statist.*, 32(1):135–166, 2004.
- A. B. Tsybakov and S. A. van de Geer. Square root penalty: adaptation to the margin in classification and in edge estimation. *Ann. Statist.*, 33, 2005.
- S. A. van de Geer. *Empirical Processes in M-Estimation*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 2000.

- A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer Series in Statistics. Springer-Verlag, New York, 1996.
- V. N. Vapnik. *Statistical Learning Theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons Inc., New York, 1998.
- V. N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer-Verlag, New York, second edition, 2000.

Convergence Theorems for Generalized Alternating Minimization Procedures

Asela Gunawardana

*Microsoft Research
One Microsoft Way
Redmond, WA 98052, U.S.A.*

ASELAG@MICROSOFT.COM

William Byrne

*University of Cambridge
Department of Engineering
Trumpington Street
Cambridge, CB2 1PZ, U.K.*

WJB31@CAM.AC.UK

Editor: Michael I. Jordan

Abstract

The EM algorithm is widely used to develop iterative parameter estimation procedures for statistical models. In cases where these procedures strictly follow the EM formulation, the convergence properties of the estimation procedures are well understood. In some instances there are practical reasons to develop procedures that do not strictly fall within the EM framework. We study EM variants in which the E-step is not performed exactly, either to obtain improved rates of convergence, or due to approximations needed to compute statistics under a model family over which E-steps cannot be realized. Since these variants are not EM procedures, the standard (G)EM convergence results do not apply to them. We present an information geometric framework for describing such algorithms and analyzing their convergence properties. We apply this framework to analyze the convergence properties of incremental EM and variational EM. For incremental EM, we discuss conditions under these algorithms converge in likelihood. For variational EM, we show how the E-step approximation prevents convergence to local maxima in likelihood.

Keywords: EM, variational EM, incremental EM, convergence, information geometry

1. Introduction

The expectation-maximization (EM) algorithm (Dempster et al., 1977) for maximum likelihood estimation (Fisher, 1922; Wald, 1949; Lehmann, 1980) is one of the most widely used parameter estimation procedures in statistical modeling. It is clear why the algorithm is attractive to researchers building statistical models. The algorithm has an elegant formulation and when it is applied to appropriate model architectures it yields parameter update procedures that are easy to derive and straightforward to implement. These parameter estimates yield increasing likelihood over the training data, and the convergence behavior of this process is well understood.

EM also has acknowledged shortcomings. It can be slow to converge or even intractable for some combinations of models and training data sets, and there are also model architectures for which the straightforward application of EM yields update procedures that do not have closed form expressions. As a result, many improvements and extensions of EM have been developed (e.g., Meilijson, 1989; Salakhutdinov et al., 2003). Incremental EM (Neal and Hinton, 1998) and vari-

ational EM (Jordan et al., 1999) are specific examples we will address in the sequel. Such extensions improve various aspects of EM, such as rate of convergence and computational tractability. However, classical (generalized) EM convergence analyses such as those of Wu (1983) and Boyles (1983) do not apply to many of these variants, and in many cases their convergence behavior is poorly understood.

We propose the generalized alternating minimization (GAM) framework with the goal of understanding the convergence properties of a broad class of such EM variants. It is based on the interpretation of EM as an alternating minimization procedure as described by Csiszár and Tusnády (1984) and later by Byrne (1992) and Amari (1995). We will show that this alternating minimization procedure can be extended in a manner analogous to the manner in which generalized EM (GEM) extends the M step of EM. We then apply a convergence argument similar to that of Wu (1983) to GAM algorithms, characterizing their convergence. This will show that GAM algorithms are a further generalization of GEM algorithms which are no longer guaranteed to increase likelihood at every iteration, but nevertheless retain convergence to stationary points in likelihood under fairly general conditions.

In practice, an iteration of EM consists of an E step which calculates sufficient statistics under the posterior distribution of the most recent model estimate, followed by an M step which generates a new model estimate from those statistics. In contrast, many variants redefine the E step to use sufficient statistics calculated under other distributions. For example, an approximation to this posterior distribution is used in variational EM (Jordan et al., 1999), and statistics from the posterior distributions of previous estimates are carried over in incremental EM (Neal and Hinton, 1998). Existing (G)EM convergence results do not apply because the E step in such variants is modified to use other “generating distributions” for computing the sufficient statistics. In order to describe such variants where the generating distribution is not necessarily the posterior distribution under the current model, GAM keeps track of both the current model and the distribution generating the statistics used for computing the next model estimate. While EM algorithms generate sequences of parameters, GAM algorithms generate sequences of parameters paired with these generating distributions.

We use the GAM framework to analyze the convergence behavior of incremental EM (Neal and Hinton, 1998) and variational EM (Jordan et al., 1999). We show that incremental EM converges to stationary points in likelihood under mild assumptions on the model family. The convergence behavior of variational EM is more complex. We do show how GAM convergence arguments can be used to guarantee the convergence of a broad class of variational EM estimation procedures. However, unlike incremental EM, this does not guarantee convergence to stationary points in likelihood. On the contrary, we show that fixed points under variational EM cannot be stationary points in likelihood, except in the degenerate case when the model family is forced to satisfy the constraints that define the variational approximation itself.

In Section 2, we review how the EM algorithm results from alternating minimization of the information divergence. First, the divergence from the current model to a family of distributions of a certain form is minimized to give a generating distribution. Then, the divergence from this distribution to the model family is minimized to give the next model. We then show that extensions of the E step such as those mentioned above involve choosing “generating distributions” that do not minimize the divergence. In GAM, the E and the M steps need only reduce—and not minimize—the divergence. In fact, the steps need not reduce the divergence individually, but may do so when applied in succession. As in EM, the modeling assumptions are represented in the parameterization of

the models. Additionally, GAM explicitly represents the approximations used in estimation by imposing constraints on the generating distributions. In pursuing this formulation we were influenced by the work of Neal and Hinton (1998) which uses generating distributions to introduce several EM variants. Our intention is to extend their analysis and provide convergence results for the algorithms they and others propose.

Understanding the convergence behavior of these variants requires the analysis of joint sequences of both parameters and their corresponding generating distributions. In Section 3 we present such an analysis. Our main convergence theorem gives conditions under which GAM procedures converge to EM fixed points. We draw on the previous work of Wu (1983) which uses results from nonlinear programming to give conditions under which (G)EM procedures converge to stationary points in likelihood, as well as the work of Csiszár and Tusnády (1984) which gives an information geometric treatment of (G)EM procedures as generating joint sequences of generating distributions and parameters. Csiszár and Tusnády (1984) also provide a convergence analysis that complements the original results of Wu (1983). However neither of the approaches generalize to EM extensions that extend the E step.

In Section 4 we apply our convergence results to incremental EM and show that although the algorithm is non-monotonic in likelihood, it does converge to EM fixed points under very general conditions. Note that Neal and Hinton (1998) have already shown that incremental EM gives non-increasing divergence (non-decreasing free energy) and that local minima in divergence (local maxima in free energy) are local maxima in likelihood. However, as we show in Section 3, this is insufficient to conclude that incremental EM converges to local maxima in likelihood, and the further analysis that is necessary is presented here. In Section 5 we apply a similar analysis to variational EM to show that convergence to EM fixed points occurs only in degenerate cases. We then conclude with some discussion in Section 6.

2. EM and Generalized Alternating Minimization

We adopt the view of the EM algorithm as an alternating minimization procedure under the information geometric framework as developed by Csiszár and Tusnády (Csiszár and Tusnády, 1984; Csiszár, 1990). This framework allows an intuitive understanding of the algorithm, and is easily extended to cover many EM variants of interest. In Section 2.1, we briefly review the EM algorithm as derived within this framework to set the groundwork for the convergence analysis of later sections. In particular we show how EM can be derived as the alternating minimization of the information divergence between the model family and a set of probability distributions constrained to be concentrated on the training data. In Section 2.2, we then extend this alternating minimization framework to generalized alternating minimization (GAM) algorithms, which are EM variants that allow extensions of the E step, in addition to the M step extensions allowed by GEM algorithms. We conclude our introduction to GAM algorithms by discussing how the GAM framework is applied to algorithms of interest in Section 2.3.

2.1 EM as Alternating Minimization

The EM algorithm, when viewed as an alternating minimization procedure, minimizes a Kullback-Leibler type divergence between a *model family* (or equivalently a parameter family) and a *desired family* of probability distributions (these are the previously mentioned generating distributions).

Let the pair of random variables X and Y be related through a function mapping X to Y . That is, X is the complete random variable and Y is the incomplete, or observed, random variable (Dempster et al., 1977; McLachlan and Krishnan, 1997). Often, X is composed of an observed and a hidden part, and Y is composed of only the observed part. We adopt the “complete”/“incomplete” variable terminology of Dempster et al. (1977) rather than the “observed”/“hidden” variable terminology that is also commonly used. The model family \mathcal{P} is defined as the set of parameterized models $P_{X;\theta}$ obtained when θ ranges over the parameter family Θ . For simplicity, we make the following assumptions

- (Q1) The complete variable X is discrete-valued.
- (Q2) $p_X(x; \theta) > 0$ for all $\theta \in \Theta$ and for all values x taken on by X . That is, the support of the models does not depend on the parameter.
- (Q3) The p.d.f. $p_X(x; \theta)$ is continuous in θ .

These technical restrictions can be relaxed to allow continuous variables (Gunawardana, 2001). The difficulty faced in doing so is that continuous models assign zero probability to the training samples; Csiszár and Tusnády (1984) show how this problem can be circumvented by the introduction of an appropriate family of dominating measures.

The desired family \mathcal{D} is defined as the set of all probability distributions Q_X that assign probability one to the observation \hat{y} of Y :

$$\mathcal{D} \triangleq \{Q_X : q_Y(\hat{y}) = 1\}$$

where Q_Y is obtained by marginalizing Q_X . Thus, desired distributions $Q_X \in \mathcal{D}$ have the property that $Q_X = Q_{X|Y=\hat{y}}$. These probability distributions are “desired” in the sense that they exemplify the maximum likelihood estimation criterion by assigning the highest possible probability to the observed data \hat{y} . Note that multiple training examples are treated by considering the sequences $X = (X^{(1)}, \dots, X^{(n)})$ and $Y = (Y^{(1)}, \dots, Y^{(n)})$ together with suitable i.i.d. assumptions.

Since we will be concerned with estimating parameterized models $P_{X;\theta}$, we define the Kullback-Leibler information divergence (Liese and Vajda, 1987) between a desired distribution $Q_X \in \mathcal{D}$ and a parameter $\theta \in \Theta$ through

$$D(Q_X || P_{X;\theta}) = \sum_x q_X(x) \log \frac{q_X(x)}{p_X(x; \theta)}. \tag{1}$$

Note that the divergence is finite for all desired distributions $Q_X \in \mathcal{D}$ and all parameters $\theta \in \Theta$ because of our simplifying assumption about the support of models $P_{X;\theta}$. This implies that the divergence is continuous over all $(Q_X, \theta) \in \mathcal{D} \times \Theta$.

Csiszár and Tusnády (1984) show that the EM algorithm can be derived as alternating minimization under the information divergence, as follows (see Figure 1):

Forward Step: Find the desired distribution $Q_X^{(t+1)}$ that minimizes the divergence from the previous parameter $\theta^{(t)}$:

$$Q_X^{(t+1)} = \operatorname{argmin}_{Q_X \in \mathcal{D}} D(Q_X || P_{X;\theta^{(t)}}).$$

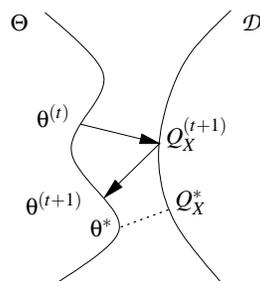


Figure 1: A schematic representation of an iteration of the alternating minimization procedure. The square of the distance between a point in Θ and a point Q_X in \mathcal{D} indicates the divergence between them. The arrows indicate projection under this divergence.

Backward Step: Find the parameter $\theta^{(t+1)}$ that minimizes the divergence to $Q_X^{(t+1)}$:

$$\theta^{(t+1)} \in \operatorname{argmin}_{\theta \in \Theta} D(Q_X^{(t+1)} || P_{X;\theta}). \quad (2)$$

In the language of Csiszár (1975), $Q_X^{(t+1)}$ is the *I-projection* of $P_{X;\theta^{(t)}}$ onto \mathcal{D} and is uniquely found as $Q_X^{(t+1)} = P_{X|Y=\hat{y};\theta^{(t)}}$. The EM algorithm (Dempster et al., 1977; Wu, 1983) can be recovered easily by substituting the I-projection into equation (2) and expanding the divergence using equation (1), to obtain

$$\theta^{(t+1)} \in \operatorname{argmax}_{\theta \in \Theta} \mathbf{E}_{P_{X|Y}} \left[\log p_X(X; \theta) \mid \hat{y}; \theta^{(t)} \right]. \quad (3)$$

Note that we use the notation $\theta^{(t+1)} \in \operatorname{argmin} D(Q_X^{(t+1)} || P_{X;\theta})$ instead of $\theta^{(t+1)} = \operatorname{argmin} D(Q_X^{(t+1)} || P_{X;\theta})$ because the backward step may not be unique.

We distinguish between the forward and backward steps of the alternating minimization procedure and the E and M steps of the EM procedure, as they are subtly different. The E step corresponds to computing the (conditional) expected log likelihood (EM auxiliary function) under the result of the forward projection. In practical implementations, the auxiliary function is not computed explicitly in the E step – the expected sufficient statistics are all that need be computed. Thus, the E step corresponds to taking an expectation under the distribution found in the forward step. The backward projection minimizes the divergence from the result of the forward projection, while the M step maximizes the expected log likelihood computed in the E step (or alternatively, finds parameters such that the sufficient statistics of the resulting model match those computed in the E step).

2.2 Generalized Alternating Minimization

There are many effective learning algorithms originally motivated by EM but which cannot be described using the formulation described above, or equivalently, using the original formulation of Dempster et al. (1977), because they generalize either the forward or the backward step. Two examples of such procedures are incremental EM (Neal and Hinton, 1998) and variational EM (Jordan

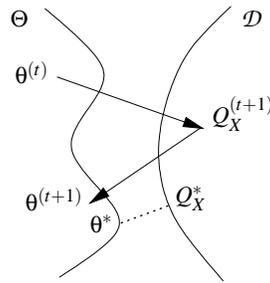


Figure 2: A schematic representation of an E step extension allowed by GAM algorithms corresponding to the M step extension of the GEM algorithm. In contrast to Figure 1, both the E and M steps reduce the divergence rather than minimizing it.

et al., 1999). We are interested in extending the alternating minimization formulation to such variants by relaxing the requirement that the forward and backward steps perform exact minimization over the families of distributions. These generalized estimation steps are described as follows.

Generalized Forward Step: Find any desired distribution $Q_X^{(t+1)}$ that reduces divergence from the previous parameter $\theta^{(t)}$:

$$Q_X^{(t+1)} : D(Q_X^{(t+1)} || P_{X;\theta^{(t)}}) \leq D(Q_X^{(t)} || P_{X;\theta^{(t)}}).$$

Generalized Backward Step: Find a parameter $\theta^{(t+1)}$ that reduces the divergence to $Q_X^{(t+1)}$:

$$\theta^{(t+1)} : D(Q_X^{(t+1)} || P_{X;\theta^{(t+1)}}) \leq D(Q_X^{(t+1)} || P_{X;\theta^{(t)}}). \tag{4}$$

Generalizations of the backward step correspond to the well known GEM algorithms. We allow similar generalization of the forward step. We refer to algorithms that consist of alternating application of such generalized forward and backward steps as generalized alternating minimization (GAM) algorithms. Thus, GAM algorithms allow for the expectation in the EM auxiliary function (equation (3)) to be found under the distribution $Q_X^{(t+1)}$ rather than $P_{X|Y=y; \theta^{(t)}}$. $Q_X^{(t+1)}$ is not chosen arbitrarily; it must be closer to $P_{X; \theta^{(t)}}$ than the desired distribution $Q_X^{(t)}$ used at the previous iteration. The effect of GAM iterations is to generate sequences of paired distributions and parameters $(Q_X^{(t)}, \theta^{(t)})$ that satisfy

$$D(Q_X^{(t+1)} || P_{X;\theta^{(t+1)}}) \leq D(Q_X^{(t)} || P_{X;\theta^{(t)}}).$$

Thus, we examine generalizations that are composed of forward and backward steps that reduce the divergence, as shown schematically in Figure 2.

2.3 Why GAM

As shown by Jordan et al. (1999), the variational EM algorithm is best described as an alternating minimization between a set of parameterized models and a set of variational approximations to the

posterior. This corresponds to extending the forward step to be a projection onto a subset of \mathcal{D} which satisfies additional constraints (namely, belonging to a given parametric family), rather than a projection onto \mathcal{D} itself.

In the following example, which follows Jordan et al. (1999), we describe how the mean field approximation to the E step arises by further constraining the desired family \mathcal{D} .

Example 1 *In the case of a Boltzmann machine, we have binary r.v.s $X = S = (S_1, \dots, S_n)$ modeled by the parametric family*

$$p_S(s; \theta) = \frac{e^{\sum_{i < j} \theta_{ij} s_i s_j + \sum_i \theta_{i0} s_i}}{z(\theta)}$$

where $z(\theta)$ ensures $P_{S; \theta}$ is properly normalized. Suppose the nodes $1, \dots, n$ of the Boltzmann machine are partitioned into a set of evidence nodes E and a set of hidden nodes H , so that $Y = S_E \triangleq (S_i)_{i \in E}$. Then, given observations \hat{s}_E of the evidence nodes, the forward step for EM estimation of the Boltzmann machine is as follows.

Forward Step: *Finding the desired distribution*

$$Q_X^{(t+1)} = \underset{Q_X \in \mathcal{D}}{\operatorname{argmin}} D(Q_X || P_{X; \theta^{(t)}})$$

gives

$$q_{S_H, S_E}^{(t+1)}(s_H, s_E) = 1_{\hat{s}_E}(s_E) p_{S_H | S_E}(s_H | \hat{s}_E; \theta^{(t)})$$

where $1_{\hat{s}_E}(s_E) = 1$ when $s_E = \hat{s}_E$ and 0 otherwise.

Note that a closed-form solution for the backward step is not generally available, but convergent algorithms can be obtained using gradient descent or iterative proportional fitting (Darroch and Ratcliff, 1972; Byrne, 1992). While the forward step can in principle be carried out exactly, this computation quickly becomes intractable as the number of states increases. In particular, direct computation of $p_{S_H | S_E}(s_H | \hat{s}_E; \theta^{(t)})$ using Bayes rule involves a summation over all possible values of the hidden nodes s_H .

To get around this we define a subset of \mathcal{D} consisting of mean field approximations to $Q_{S_H | S_E}$. That is, we define \mathcal{D}_{MF} to be those distributions in \mathcal{D} whose p.d.f. has the parametric form

$$q_S(s; \mu) = 1_{\hat{s}_E}(s_E) \prod_{i \in H} \underbrace{\mu_i^{s_i} (1 - \mu_i)^{1 - s_i}}_{q_{S_i; \mu_i}}$$

where each μ_i takes values in $[0, 1]$. Thus the members of \mathcal{D}_{MF} allow no dependencies between nodes. It follows that a distribution $Q_S \in \mathcal{D}_{MF}$ is fully specified by its parameter μ and the training observations \hat{s}_E .

The forward step can then be replaced by an approximate forward step, which is now a minimization over the variational parameter μ for fixed $\theta^{(t)}$:

Approximate Forward Step: *An (approximate) desired distribution*

$$Q_X^{(t+1)} \in \underset{Q_X \in \mathcal{D}_{MF}}{\operatorname{argmin}} D(Q_X || P_{X; \theta^{(t)}})$$

with p.d.f.

$$q_S^{(t+1)}(s) = 1_{s_E}(s_E) \prod_{i \in H} q_{S_i}(s_i; \mu_i^{(t+1)})$$

is chosen by finding a variational parameter

$$\mu^{(t+1)} \in \underset{\mu}{\operatorname{argmin}} D(Q_{S;\mu} || P_{S;\theta^{(t)}}).$$

As described by Jordan et al. (1999), this can be done directly, without needing to compute $P_{S_H|S_E;\theta^{(t)}}$, by solving the nonlinear system of mean field equations

$$\mu_i^{(t+1)} = \sigma \left(\sum_j \theta_{ij}^{(t)} \mu_j^{(t+1)} + \theta_{i0} \right),$$

where $\sigma(\cdot)$ is the logistic function. Note that this simplification results from the careful crafting of the parametric form imposed on \mathcal{D}_{MF} .

It can be seen that this variational EM variant is easily described in terms of minimizing the divergence between a constrained family of desired distributions and a model family. The approximate forward step in this example is a generalization of the usual I-projection onto \mathcal{D} , and the resulting algorithm is therefore a GAM procedure.

3. GAM Convergence

In this section, we describe our main result – a theorem which characterizes the convergence of GAM procedures. As the preceding example shows, some EM variants of interest are GAM procedures but not GEM procedures. This means that their convergence behavior may be different from what the familiar convergence properties of (G)EM would suggest. In particular, monotonic increase in likelihood and convergence to local maxima (technically, stationary points) in likelihood may no longer hold. This may happen even when the divergence is non-increasing, and when stationary points of the likelihood are fixed points of the GAM procedure. We begin with a simple toy example where this can easily be seen.

Example 2 Let the complete random variable $X = (X_1, X_2)$ represent the result of tossing a coin twice. That is, X_1, X_2 are i.i.d., with X_i taking the value 1 with probability θ and 0 with probability $1 - \theta$. Let the incomplete random variable Y encode whether the result seemed “fair” or not. It takes on the value 1 if X takes on the values (0,1) or (1,0), and takes on the value 0 otherwise. Suppose the observation \hat{y} of Y is $\hat{y} = 0$. In this simple case, the complete data likelihood is given by

$$p_X(x; \theta) = \theta^{x_1+x_2} (1 - \theta)^{2-(x_1+x_2)}$$

and the incomplete data likelihood is given by

$$\begin{aligned} p_Y(\hat{y}; \theta) &= p_Y(0; \theta) \\ &= p_X(0,0; \theta) + p_X(1,1; \theta) \\ &= (1 - \theta)^2 + \theta^2. \end{aligned}$$

Note that the incomplete likelihood is convex, with global maxima at $\theta = 0$ and $\theta = 1$, and a global minimum at $\theta = 0.5$. Desired distributions $Q_X \in \mathcal{D}$ take the form

$$q_X(x) = \begin{cases} q_{11} & \text{if } x = (1, 1), \\ 1 - q_{11} & \text{if } x = (0, 0), \\ 0 & \text{otherwise.} \end{cases}$$

The divergence between a desired distribution and a model is given by

$$D(Q_X || P_{X;\theta}) = q_{11} \log \frac{q_{11}}{\theta^2} + (1 - q_{11}) \log \frac{1 - q_{11}}{(1 - \theta)^2},$$

which can be shown to be convex in q_{11} for fixed θ and convex in θ for fixed q_{11} (though not jointly convex in q_{11} and θ). The EM algorithm for estimating θ can be given by a forward step and a backward step as follows:

Forward Step: As described above, the forward step is given by the I-projection of the model $P_{X;\theta^{(t)}}$ onto \mathcal{D} . This is given by

$$q_{X|Y}^{(t+1)}(x|\hat{y}) = p_{X|Y}(x|\hat{y}; \theta^{(t)}),$$

$$q_{11}^{(t+1)} = \frac{\theta^{(t)^2}}{(1 - \theta^{(t)})^2 + \theta^{(t)^2}}.$$

Backward Step: Minimizing the divergence given above over θ for a fixed q_{11} gives

$$\theta^{(t+1)} = q_{11}^{(t+1)}.$$

Thus, the EM iteration for this problem is

$$\theta^{(t+1)} = \frac{\theta^{(t)^2}}{(1 - \theta^{(t)})^2 + \theta^{(t)^2}}.$$

It can be seen that $\theta^{(0)} < 0.5$ gives convergence to the global maximum at $\theta = 0.0$, while $\theta^{(0)} > 0.5$ gives convergence to the global maximum at $\theta = 1.0$. Starting at the global minimum at $\theta = 0.5$ traps the algorithm there.

We now investigate how the additional constraint

$$0.4 \leq q_{11} \leq 0.6 \tag{5}$$

on the desired distribution changes the forward step, and as a result, the convergence behavior of the algorithm. Note that a forward step that projects onto this constrained set of desired distributions will reduce the divergence between the desired distribution and the model, and will therefore be a GAM procedure.

Computing the partial derivative

$$\frac{\partial}{\partial q_{11}} D(Q_X || P_{X;\theta}) = \log \left(\frac{q_{11}}{1 - q_{11}} \left(\frac{1 - \theta}{\theta} \right)^2 \right)$$

shows that it is positive for $0.4 \leq q_{11} \leq 0.6$ when $\theta < \frac{1}{1+\sqrt{3/2}} \approx 0.4495$. Therefore, the forward step from any $\theta < \frac{1}{1+\sqrt{3/2}}$ is given by $q_{11} = 0.4$.

Suppose $\theta^{(0)} = 0.3$. The unconstrained forward step would have given $q_{11}^{(1)} = 0.155$, which would have violated the additional constraint (5). Under the additional constraint (5), the forward step is given by $q_{11}^{(1)} = 0.4$. This in turn leads to $\theta^{(1)} = 0.4$, and the next forward step again gives $q_{11}^{(2)} = 0.4$, showing that the algorithm has converged in a single step, albeit to a value that is not a maximum (or stationary point) in likelihood. Also, recall that the incomplete data likelihood is convex with a minimum at $\theta = 0.5$. This means that initial points in $\theta < 0.4$ will converge in one step to $\theta = 0.4$, thereby reducing likelihood. Indeed, the likelihood at the initial point ($\theta = 0.3$) is 0.58 and the likelihood at the subsequent (limit) points ($\theta = 0.4$) is 0.52.

Thus, it is clear that the convergence behavior of GAM algorithms can differ extremely from that of EM algorithms, and therefore needs to be carefully studied. In fact, non-monotonic convergence behavior can also be seen in the case of incremental EM (Byrne and Gunawardana, 2000). In the following, we will show that under smoothness conditions on the forward and backward steps, GAM procedures that strictly reduce divergence at every step, except possibly at stationary points in likelihood, will yield solutions that are stationary points in likelihood.

3.1 GAM Convergence Theorem

The GAM convergence theorem is a direct application of the generalized convergence theorem (GCT) of Zangwill (1969). We will define the forward and the backward steps to be point-to-set maps, rather than functions, so that we may deal with extended E and M steps that do not yield unique iterates. The GCT will require that these maps be closed. Closedness of a point-to-set map is a smoothness property that is related to function continuity, and is defined as follows:

Definition 1 A point-to-set-map $H : U \rightarrow V$ is closed at $u \in U$ if for any two sequences $\{u^{(t)}\}_{t=0}^\infty \in U$ and $\{v^{(t)}\}_{t=0}^\infty \in V$ the conditions $u^{(t)} \rightarrow u$, $v^{(t)} \rightarrow v$, and $v^{(t)} \in H(u^{(t)})$, imply that $v \in H(u)$.

We now state Zangwill (1969)'s GCT:

Theorem 2 Let the point-to-set map $H : Z \rightarrow Z$ determine an algorithm that given a point $z^{(0)}$ generates a sequence $\{z^{(t)}\}_{t=0}^\infty$ through the iteration $z^{(t+1)} \in H(z^{(t)})$. Also let a solution set Γ be given. Suppose

- (1) All points $z^{(t)}$ are in a compact set $S \subseteq Z$.
- (2) There is a continuous function $\alpha : Z \rightarrow \mathbb{R}$ such that:
 - (a) if $z \notin \Gamma$, then $\alpha(z') < \alpha(z) \forall z' \in H(z)$,
 - (b) if $z \in \Gamma$, then $\alpha(z') \leq \alpha(z) \forall z' \in H(z)$.
- (3) The map H is closed at z if $z \notin \Gamma$.

Then the limit of any convergent subsequence of $\{z^{(t)}\}_{t=0}^\infty$ is in Γ . That is, accumulation points z^* of the sequence $z^{(t)}$ lie in Γ . Furthermore, $\alpha(z^{(t)})$ converges to α^* , and $\alpha(z^*) = \alpha^*$ for all accumulation points z^* .

We use this GCT to show our main convergence result for GAM procedures and then give a corollary that describes how they converge in likelihood.

Theorem 3 (GAM Convergence Theorem) *Let \mathcal{D} be any family of distributions on X and let Θ be the parameter family defined in Section 2. Let the solution set Γ be defined as*

$$\Gamma = \left\{ (Q_X, \theta) : Q_X \in \underset{Q'_X \in \mathcal{D}}{\operatorname{argmin}} D(Q'_X || P_{X;\theta}) \text{ and } \theta \in \underset{\xi \in \Theta}{\operatorname{argmin}} D(Q_X || P_{X;\xi}) \right\}.$$

Let $FB : \mathcal{D} \times \Theta \rightarrow \mathcal{D} \times \Theta$ be any point-to-set map such that all $(Q'_X, \theta') \in FB(Q_X, \theta)$ satisfy

$$(GAM) : \quad D(Q'_X || P_{X;\theta'}) \leq D(Q_X || P_{X;\theta})$$

with equality only if

$$(EQ) : \quad (Q_X, \theta) \in \Gamma.$$

Let $\{(Q_X^{(t)}, \theta^{(t)})\}_{t=0}^{\infty} \in \mathcal{D} \times \Theta$ be a sequence generated from a pair $(Q_X^{(0)}, \theta^{(0)})$ by the iterative application of the point-to-set-map FB :

$$(Q_X^{(t+1)}, \theta^{(t+1)}) \in FB(Q_X^{(t)}, \theta^{(t)}).$$

Suppose that Θ is compact, that there is a compact set $\mathcal{D}' \subseteq \mathcal{D}$ such that

$$(1) \quad FB(\mathcal{D}' \times \Theta) \stackrel{\Delta}{=} \cup_{(Q_X, \theta) \in \mathcal{D}' \times \Theta} FB(Q_X, \theta) \subseteq \mathcal{D}' \times \Theta,$$

(2) The point-to-set map FB is closed on $\mathcal{D}' \times \Theta$,

and that it can be shown that $(Q_X^{(k)}, \theta^{(k)}) \in \mathcal{D}' \times \Theta$ for some iteration (k) .

Then all accumulation points (Q_X^, θ^*) of the sequence $\{(Q_X^{(t)}, \theta^{(t)})\}_{t=0}^{\infty}$ lie in the solution set Γ and $D(Q_X^* || P_{X;\theta^*}) = D^*$ and $D(Q_X^{(t)} || P_{X;\theta^{(t)}}) \rightarrow D^*$.*

Proof We restrict the point-to-set map FB to $\mathcal{D}' \times \Theta$, and then apply Zangwill's GCT above with $S = Z = \mathcal{D}' \times \Theta$, $\alpha = D$, $H = FB$, and $\{z^{(t)}\}_{t=0}^{\infty} = \{(Q_X^{(t)}, \theta^{(t)})\}_{t=k}^{\infty}$. The compactness of $\mathcal{D}' \times \Theta$ follows from the compactness of \mathcal{D}' and Θ individually. The continuity of the divergence in (Q_X, θ) follows from the continuity of the divergence $D(Q_X || P_{X;\theta})$ in Q_X and $P_{X;\theta}$ and the continuity of $p_{X;\theta}$ in θ . The theorem then follows by direct application of Zangwill's theorem. ■

Corollary 4 (Stationary Points in Likelihood) *In Theorem 3, suppose that \mathcal{D} is the desired family defined in Section 2. Then the following hold for accumulation points (Q_X^*, θ^*) :*

$$(1) \quad p_Y(\hat{y}; \theta^*) = e^{-D^*} \text{ and } p_Y(\hat{y}; \theta^{(t)}) \rightarrow e^{-D^*}.$$

(2) θ^ is a stationary point of the incomplete data likelihood if it is in the interior of Θ .*

Proof For $(Q_X, \theta) \in \Gamma$, $q_X(x) = p_{X|Y}(x|\hat{y}; \theta)$ so that $D(Q_X || P_{X; \theta}) = -\log q(\hat{y}; \theta)$ yielding conclusion (1).

Since $(Q_X^*, \theta^*) \in \Gamma$, $q_X^*(x) = p_{X|Y}(x|\hat{y}; \theta^*)$, giving

$$\theta^* \in \arg \min_{\theta \in \Theta} D(P_{X|Y=\hat{y}; \theta^*} || P_{X; \theta}).$$

The divergence in the right hand side can be expanded as

$$D(P_{X|Y=\hat{y}; \theta^*} || P_{X; \theta}) = -\log p_Y(\hat{y}; \theta) + D(P_{X|Y=\hat{y}; \theta^*} || P_{X|Y=\hat{y}; \theta}).$$

Taking the gradient of this expression and setting it to zero yields

$$-\nabla_{\theta} \log p_Y(\hat{y}; \theta) \Big|_{\theta=\theta^*} + \nabla_{\theta} D(P_{X|Y=\hat{y}; \theta^*} || P_{X|Y=\hat{y}; \theta}) \Big|_{\theta=\theta^*} = 0.$$

Since $D(P_{X|Y=\hat{y}; \theta^*} || P_{X|Y=\hat{y}; \theta})$ is minimized when $\theta = \theta^*$, this gives us that

$$\nabla_{\theta} \log p_Y(\hat{y}; \theta) \Big|_{\theta=\theta^*} = 0.$$

This proves conclusion (2). ■

The GAM convergence theorem and corollary provide conditions under which iterative estimation procedures converge to stationary points in likelihood. However it is possible that these procedures are not monotonic in likelihood. This can be seen from the Pythagorean equality (Csiszár, 1975) which provides the following relationship between all Q_X in the linear family \mathcal{D} and a model $P_{X; \theta}$

$$D(Q_X || P_{X; \theta}) = D(Q_X || \tilde{Q}_X) + D(\tilde{Q}_X || P_{X; \theta})$$

where the I-projection $\tilde{Q}_X = \operatorname{argmin}_{Q_X \in \mathcal{D}} D(Q_X || P_{X; \theta})$ is uniquely specified as $\tilde{Q}_{X|Y=\hat{y}} = P_{X|Y=\hat{y}; \theta}$. From this we find the following relationship between the likelihood of the model estimates and the overall divergence

$$D(Q_X || P_{X; \theta}) = D(Q_X || P_{X|Y=\hat{y}; \theta}) - \log p_Y(\hat{y}; \theta).$$

While GAM procedures guarantee that $D(Q_X^{(t+1)} || P_{X; \theta^{(t+1)}}) \leq D(Q_X^{(t)} || P_{X; \theta^{(t)}})$, we can conclude only that

$$\log p_Y(\hat{y}; \theta^{(t+1)}) \geq \log p_Y(\hat{y}; \theta^{(t)}) + \Delta^{(t)}$$

where $\Delta^{(t)} = D(Q_X^{(t+1)} || P_{X|Y=\hat{y}; \theta^{(t+1)}}) - D(Q_X^{(t)} || P_{X|Y=\hat{y}; \theta^{(t)}})$. Since, as shown in Figure 3, this quantity can be negative, it is possible for GAM algorithms to be non-monotonic in likelihood even while converging to local maxima in likelihood.

We now discuss the construction of a GAM mapping FB that satisfies the requirements of the GAM convergence theorem.

Proposition 5 *Let the point-to-set map FB in Theorem 3 above be the composition $B \circ F$ of point-to-set maps $F : \mathcal{D} \times \Theta \rightarrow \mathcal{D} \times \Theta$ and $B : \mathcal{D} \times \Theta \rightarrow \mathcal{D} \times \Theta$. Suppose that the point-to-set maps F and B are defined so that*

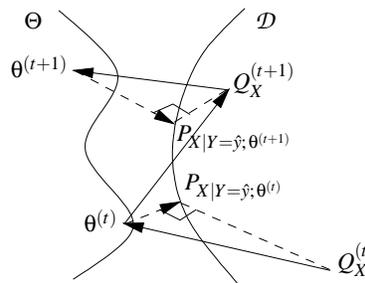


Figure 3: A schematic representation of how GAM procedures may be non-monotonic in likelihood. The solid arrows show forward and backward steps that reduce the divergence rather than minimizing it. The broken arrows show the forward steps that would have been taken by the EM algorithm (i.e., the I-projections of the models). Divergences that obey the Pythagorean equality are indicated by right triangles. In particular, the squared lengths of the broken arrows represent negative log likelihood. Note that the divergence between the desired distribution yielded by the forward step and the I-projection of the model decreases, while the negative log likelihood increases.

(1) F and B are closed on $\mathcal{D}' \times \Theta$

(2) $F(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$ and $B(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$

Suppose also that F is such that all $(Q'_X, \theta') \in F(Q_X, \theta)$ have $\theta' = \theta$ and satisfy

$$(GAM.F) : \quad D(Q'_X || P_{X;\theta}) \leq D(Q_X || P_{X;\theta})$$

with equality only if

$$(EQ.F) : \quad Q_X = \operatorname{argmin}_{Q'_X \in \mathcal{D}} D(Q'_X || P_{X;\theta}),$$

with Q_X being the unique minimizer. Suppose also that the point-to-set map B is such that all $(Q'_X, \theta') \in B(Q_X, \theta)$ have $Q'_X = Q_X$ and satisfy

$$(GAM.B) : \quad D(Q_X || P_{X;\theta'}) \leq D(Q_X || P_{X;\theta})$$

with equality only if

$$(EQ.B) : \quad \theta \in \operatorname{argmin}_{\xi \in \Theta} D(Q_X || P_{X;\xi}).$$

Then,

(1) the point-to-set map FB is closed on $\mathcal{D}' \times \Theta$

(2) $FB(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$

and FB satisfies the GAM and EQ conditions of the GAM convergence theorem.

Proof If the point-to-set maps $F : A \rightarrow B$ and $G : B \rightarrow C$ are closed on A and B respectively, their composition $FG = G \circ F$ is closed on A if B is compact. Since F and B are closed on $\mathcal{D}' \times \Theta$, which is compact, it follows that FB is closed on $\mathcal{D}' \times \Theta$. That $FB(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$ follows directly from the assumptions of the proposition.

The condition (GAM) follows directly from (GAM.F) and (GAM.B).

Conditions (EQ.F) and (EQ.B) together are not enough to ensure condition (EQ). Suppose $(R_X, \phi) \in FB(Q_X, \theta)$. This implies that $(R_X, \theta) \in F(Q_X, \theta)$ and $(R_X, \phi) \in B(R_X, \theta)$.

Suppose $D(R_X || P_{X;\phi}) = D(Q_X || P_{X;\theta})$. Then (GAM.F) and (GAM.B) ensure that $D(R_X || P_{X;\phi}) = D(R_X || P_{X;\theta}) = D(Q_X || P_{X;\theta})$. Condition (EQ.F) gives

$$\begin{aligned} Q_X &\in \arg \min_{Q'_X \in \mathcal{D}} D(Q'_X || P_{X;\theta}), \\ R_X &\in \arg \min_{Q'_X \in \mathcal{D}} D(Q'_X || P_{X;\theta}), \end{aligned} \tag{6}$$

and (EQ.B) gives

$$\theta \in \arg \min_{\xi \in \Theta} D(R_X || P_{X;\xi}). \tag{7}$$

While equation (6) is the first criterion for membership in Γ , equation (7) is not quite the second criterion – the divergence minimized here is $D(R_X || P_{X;\xi})$ instead of $D(Q_X || P_{X;\xi})$. Since by assumption, Q_X is the unique minimizer of the divergence, $Q_X = R_X$, giving the required condition

$$\theta \in \arg \min_{\xi \in \Theta} D(Q_X || P_{X;\xi}).$$

■

This allows us to construct a map FB through the composition of generalized forward and backward steps F and B . As seen in the proof it is insufficient for the forward and backward steps to satisfy the GAM and EQ conditions separately. It is also necessary for the forward step to satisfy the equality condition with a unique minimizer. For example, this condition is satisfied when \mathcal{D} is defined by linear constraints as in Section 2 and the forward step is a simple projection, as in the case of EM. Even when this condition is not satisfied, it may be possible to show condition (EQ) for the composite map FB . It is important to show that FB strictly decreases the divergence for all points outside the solution set Γ , since any points where this does not hold are accumulation points of the algorithm.

As an instance of the GAM procedure, EM convergence is also explained by these results as shown in Appendix A. The conditions of Theorem 3 and Corollary 4 are quite general, and very similar to those that must be satisfied to ensure GEM convergence (Wu, 1983). For example, in both GEM and GAM, condition (Q2) must hold. Insisting on this would rule out GMMs with parameter families that allow individual Gaussians to have a variance of zero. In practice, modeling considerations usually prevent such situations.

4. Incremental EM as GAM

We now turn our attention to the incremental EM algorithm of Neal and Hinton (1998). This variant of the EM algorithm divides the training data into partitions, and at each iterate, computes conditional sufficient statistics on only one partition. The statistics conditioned on other partitions are

saved from previous iterations. The statistics corresponding to the different partitions are pooled before performing the M step at each iteration, but the separate per-partition statistics are retained for use in future iterations. This algorithm has shown to give faster convergence in a number of applications (Digalakis, 1997; Thiesson et al., 2001; Hsiao et al., 2004), even though it may be non-monotonic in likelihood (Byrne and Gunawardana, 2000). Here, we use our GAM results to show that in most cases, the incremental updates do not sacrifice the convergence guarantees of EM, despite the non-monotonicity in likelihood. Note that Neal and Hinton (1998) have shown that incremental EM is monotonic in divergence, but not that it converges to EM fixed points.

The complete variable $X = (X^{(1)}, \dots, X^{(n)})$ is assumed to consist of n independent components so that $Q_X = \prod_{i=1}^n Q_{X^{(i)}}$. The visible variable $Y = (Y^{(1)}, \dots, Y^{(n)})$ has observed value $\hat{y} = (\hat{y}^{(1)}, \dots, \hat{y}^{(n)})$. The components $Y^{(i)}$ are generated independently of each other, from their corresponding $X^{(i)}$.

The EM auxiliary function for these variables is

$$\begin{aligned} \Phi(\theta|\theta^{(t)}) &= \sum_{i=1}^n \mathbf{E}_{P_{X^{(i)}|Y^{(i)}}} \left[\log p_{X^{(i)}}(X^{(i)}; \theta) \mid \hat{y}^{(i)}; \theta^{(t)} \right] \\ &= \sum_{i=1}^n \Phi^{(i)}(\theta|\theta^{(t)}). \end{aligned}$$

Rather than maximize this auxiliary function, the incremental EM algorithm allows re-estimation to be performed based on a single component $\hat{y}^{(i)}$ of the observation \hat{y} at any step. For example, in a two-element problem the re-estimation procedure might proceed as follows :

$$\begin{aligned} \theta^{(t+1)} &= \operatorname{argmax}_{\theta \in \Theta} (\Phi^{(1)}(\theta|\theta^{(t-1)}) + \Phi^{(2)}(\theta|\theta^{(t)})), \\ \theta^{(t+2)} &= \operatorname{argmax}_{\theta \in \Theta} (\Phi^{(1)}(\theta|\theta^{(t+1)}) + \Phi^{(2)}(\theta|\theta^{(t)})), \\ \theta^{(t+3)} &= \operatorname{argmax}_{\theta \in \Theta} (\Phi^{(1)}(\theta|\theta^{(t+1)}) + \Phi^{(2)}(\theta|\theta^{(t+2)})), \\ &\dots \end{aligned}$$

This is not enough to ensure that $\Phi(\theta^{(t+3)}|\theta^{(t+1)}) \leq \Phi(\theta^{(t+1)}|\theta^{(t+1)})$ so the (G)EM convergence results do not apply. However the algorithm can be formulated as an GAM procedure.

To show that incremental EM can be a GAM procedure, we describe it as a nested series of n incremental forward steps and n exact backward steps. Iteration $(t+1)$ of incremental EM proceeds as follows. First, the iteration is initialized from the results of the previous iteration:

$$Q_X^{(t+1,0)} = Q_X^{(t)} \text{ and } \theta^{(t+1,0)} = \theta^{(t)}.$$

We then define a series of n incremental forward steps $j = 1, \dots, n$

$$Q_{X^{(i)}}^{(t+1,j)} = \begin{cases} P_{X^{(i)}|Y^{(i)}=\hat{y}^{(i)}; \theta^{(t+1,j-1)}} & \text{if } j = i \\ Q_{X^{(i)}}^{(t+1,j-1)} & \text{otherwise,} \end{cases}$$

and backward steps

$$\theta^{(t+1,j)} \in \operatorname{argmin}_{\xi \in \Theta} D(Q_X^{(t+1,j)} \| P_{X;\xi}),$$

so that finally we set $Q_X^{(t+1)} = Q_X^{(t+1,n)}$ and $\theta^{(t+1)} = \theta^{(t+1,n)}$.

We formally represent the (j) th incremental forward step $F^{(j)} : \mathcal{D} \times \Theta \rightarrow \mathcal{D} \times \Theta$ as the singleton point-to-set map

$$F^{(j)}(Q_X, \theta) = \left\{ (Q'_X, \theta') : Q'_X = P_{X^{(j)}|Y^{(j)}=\hat{y}^{(j)}; \theta} \prod_{i \neq j} Q_{X^{(i)}} \right\}.$$

It updates the (j) th component marginal $Q_{X^{(j)}}$ of Q_X but keeps the other component marginals fixed.

The backward step is represented by a closed point-to-set map $B : \mathcal{D} \times \Theta \rightarrow \mathcal{D} \times \Theta$ satisfying conditions (GAM.B) and (EQ.B) of Proposition 5 with $Q'_X = Q_X$ for $(Q'_X, \theta') \in B(Q_X, \theta)$, and additionally satisfying:

$$B(Q_X, \theta) \text{ is a singleton set } \forall (Q_X, \theta) \in \mathcal{D} \times \Theta : (Q_X, \theta) \in M(Q_X, \theta). \quad (8)$$

Thus, we are guaranteed that $\theta' = \theta$ when $D(Q_X || P_{X; \theta'}) = D(Q_X || P_{X; \theta})$. This is equivalent to requiring that the EM auxiliary function has a unique maximizer. We note that this often holds in practice – for example, when the complete data distribution comes from a flat exponential family (Efron, 1975; Amari, 1995) as is the case with mixtures of Gaussians, or with hidden Markov models. Even when the complete data distribution is a curved exponential family, uniqueness can still be possible.

Using these composite maps we can describe incremental EM as

$$(Q_X^{(t+1)}, \theta^{(t+1)}) \in FB(Q_X^{(t)}, \theta^{(t)})$$

where

$$FB = B \circ F^{(n)} \circ \dots \circ B \circ F^{(1)}.$$

Proposition 6 *As defined, incremental EM can be shown to converge to stationary points in likelihood through application of the GAM convergence theorem.*

Proof For any $(Q_X, \theta) \in \mathcal{D} \times \Theta$, we use the independence of the components $X^{(i)}$ and $Y^{(i)}$ to decompose the divergence $D(Q_X || P_{X; \theta})$ into a sum of component divergences as follows

$$D(Q_X || P_{X; \theta}) = \sum_i D(Q_{X^{(i)}} || P_{X^{(i)}; \theta}).$$

The (j) th backward step satisfies

$$\begin{aligned} D(Q_X^{(t+1,j)} || P_{X; \theta^{(t+1,j)}}) &\leq D(Q_X^{(t+1,j)} || P_{X; \theta^{(t+1,j-1)}}) \\ &= \sum_{i: i \neq j} D(Q_{X^{(i)}}^{(t+1,j-1)} || P_{X; \theta^{(t+1,j-1)}}) + D(Q_{X^{(j)}}^{(t+1,j)} || P_{X; \theta^{(t+1,j-1)}}) \end{aligned}$$

where the right hand side has been expanded using the fact that the (j) th incremental forward step leaves all but the (j) th component divergence unchanged. Since the (j) th incremental forward step minimizes the (j) th component divergence, we get

$$\begin{aligned} D(Q_X^{(t+1,j)} || P_{X; \theta^{(t+1,j)}}) &\leq \sum_{i: i \neq j} D(Q_{X^{(i)}}^{(t+1,j-1)} || P_{X; \theta^{(t+1,j-1)}}) + D(Q_{X^{(j)}}^{(t+1,j-1)} || P_{X; \theta^{(t+1,j-1)}}) \\ &= D(Q_X^{(t+1,j-1)} || P_{X; \theta^{(t+1,j-1)}}). \end{aligned}$$

Condition (GAM) of Theorem 3 is therefore satisfied.

Since the maps $F^{(j)}$ and B are closed (Appendix A, Proposition 7), FB will be closed on \mathcal{D}' , if the set is constructed so as to be compact (Appendix A). An appropriate definition of $\mathcal{D}' \subseteq \mathcal{D}$ is given in Appendix B along with a proof that incremental EM satisfies the equality condition (EQ) of Theorem 3. ■

Thus, the GAM convergence theorem shows that incremental EM procedures converge to EM fixed points when the EM auxiliary function is uniquely maximized. However, it is not a GEM procedure, and monotonicity in likelihood is no longer guaranteed. Indeed, as discussed in Byrne and Gunawardana (2000) non-monotonicity in likelihood is observed in practice, and the convergence behavior is very different from that of (G)EM procedures, despite the common fixed point set. Thiesson et al. (2001) also show that the convergence behavior of incremental EM is different from that of EM in practice.

5. Variational EM as GAM

Variational approximations have been popular in cases where computing the exact forward step $q_X(x) = p_{X|Y}(x|\hat{y}; \theta)$ is intractable (Jordan et al., 1999). The idea is to restrict attention to a subfamily \mathcal{D}_V of \mathcal{D} such that members of \mathcal{D}_V have a particular parametric form, which is chosen so that projecting a model $P_{X; \theta}$ onto \mathcal{D}_V is more tractable than projecting it onto \mathcal{D} . That is, a parametrization $q_X(x; \lambda)$ with $\lambda \in \Lambda$ is fixed, and the family \mathcal{D}_V is defined as

$$\mathcal{D}_V = \{Q_X \in \mathcal{D} : q_X(x) = q_X(x; \lambda) \text{ for some } \lambda \in \Lambda\}.$$

We assume $\Lambda \subseteq \mathbb{R}^n$ is closed and bounded.

Then, the variational forward step is defined to be

$$F_V(Q_X, \theta) = \left\{ (Q'_X, \theta) : Q'_X \in \arg \min_{Q''_X \in \mathcal{D}_V} D(Q''_X \| P_{X; \theta}) \right\}.$$

By the Pythagorean equality of Csiszár (1975),

$$\begin{aligned} D(Q''_X \| P_{X; \theta}) &= D(Q''_X \| P_{X|Y=\hat{y}; \theta}) + D(P_{X|Y=\hat{y}; \theta} \| P_{X; \theta}) \\ &= D(Q''_X \| P_{X|Y=\hat{y}; \theta}) - \log p_Y(\hat{y}; \theta). \end{aligned}$$

Thus, $Q''_X \in \mathcal{D}_V$ that minimizes this divergence also best approximates $P_{X|Y=\hat{y}; \theta}$, which is the desired distribution that would be chosen by the usual EM procedure.

Notice that the divergence minimized at every iteration is no longer just $D(P_{X|Y=\hat{y}; \theta} \| P_{X; \theta})$ (which is the negative log likelihood) as in the EM algorithm, and that therefore, the likelihood is not guaranteed to increase at every iteration. We now examine if the conditions of the GAM convergence theorem of Section 3 still hold if the forward step of the EM procedure is replaced by F_V .

First, note that \mathcal{D}_V is a natural choice for \mathcal{D}' as long as the set of variational parameters Λ is compact. That the map F_V is closed on $\mathcal{D}_V \times \Theta$ follows from Corollary 8 and Lemma 9 of Appendix A, and the assumptions on Λ . The mapping satisfies conditions (GAM.F) and (EQ.F) because each new desired distribution must minimize the divergence to \mathcal{D}_V . However, the uniqueness condition

of Proposition 5 (EQ.F) cannot be guaranteed in general, and must be verified for each choice of \mathcal{D}_V . If this condition holds, then the algorithm converges to minimizers of the divergence between the family of variational approximations and the model family. For example, this happens when the variational E step is uniquely defined.

We now analyze when these limit points (Q_X^*, θ^*) are stationary points in likelihood. Since θ^* minimizes the divergence $D(Q^* || P_{X; \theta})$ over θ ,

$$\nabla_{\theta} D(Q_X^* || P_{X; \theta}) \Big|_{\theta=\theta^*} = 0.$$

Expanding the divergence as before,

$$\nabla_{\theta} D(Q_X^* || P_{X|Y=\hat{y}; \theta}) \Big|_{\theta=\theta^*} - \nabla_{\theta} \log q(\hat{y}; \theta) \Big|_{\theta=\theta^*} = 0$$

so that $\nabla_{\theta} \log q(\hat{y}; \theta) \Big|_{\theta=\theta^*} = 0$ if and only if

$$\nabla_{\theta} D(Q_X^* || P_{X|Y=\hat{y}; \theta}) \Big|_{\theta=\theta^*} = 0.$$

Therefore a θ^* generated by a variational EM procedure is a stationary point in likelihood if and only if θ^* is a parameter that locally minimizes the variational approximation error. This can happen in two ways. First, the variational error may have stationary points at stationary points in likelihood. This can only be ensured if the stationary points are known before estimation. Second, the variational error is independent of θ . This is not possible if the variational family introduces independence assumptions that ensure tractability. In particular, a model which agrees with the variational approximation (e.g., a factorial HMM with parameter settings that decouple the state sequences) will have lower variational error than one that does not. We illustrate this in the case of the mean field approximation for Boltzmann machines.

Example 3 *In Example 1, choose a pair of hidden nodes i, j connected by a dependency link. It is well-known (Byrne, 1992) that*

$$\begin{aligned} \frac{\partial}{\partial \theta_{ij}} \log P_{S|S_E}(s|\hat{s}_E; \theta) &= s_i s_j - \mathbf{E}_{P_{S|S_E}} [S_i S_j | \hat{s}_E; \theta], \\ \frac{\partial}{\partial \theta_{k0}} \log P_{S|S_E}(s|\hat{s}_E; \theta) &= s_k - \mathbf{E}_{P_{S|S_E}} [S_k | \hat{s}_E; \theta] \quad k = i, j, \end{aligned}$$

which gives

$$\begin{aligned} \frac{\partial}{\partial \theta_{ij}} D(Q_S^* || P_{S|S_E=\hat{s}_E; \theta}) &= \mathbf{E}_{Q_S^*} [S_i S_j; \mu] - \mathbf{E}_{P_{S|S_E}} [S_i S_j | \hat{s}_E; \theta] \\ &= \mu_i^* \mu_j^* - \mathbf{E}_{P_{S|S_E}} [S_i S_j | \hat{s}_E; \theta], \\ \frac{\partial}{\partial \theta_k} D(Q_S^* || P_{S|S_E=\hat{s}_E; \theta}) &= \mu_k^* - \mathbf{E}_{P_{S|S_E}} [S_k | \hat{s}_E; \theta] \quad k = i, j. \end{aligned}$$

If $\nabla_{\theta} D(Q_S^* || P_{S|S_E=\hat{s}_E}; \theta) \Big|_{\theta=\theta^*}$ is to be zero,

$$\mathbf{E}_{P_{S|S_E}} [S_i S_j | \hat{s}_E; \theta^*] = \mathbf{E}_{P_{S|S_E}} [S_i | \hat{s}_E; \theta^*] \mathbf{E}_{P_{S|S_E}} [S_j | \hat{s}_E; \theta^*]$$

must hold. This can only occur if $\theta_{ij}^* = 0$, when the model itself satisfies the constraints of the mean field approximation. Thus, under this variational approximation, only Boltzmann machines where the hidden units do not depend on each other can give stationary points in likelihood.

To summarize, the GAM convergence theorem applies to variational EM in cases when the variational E step is uniquely defined. When this is so, the resulting model is at a local minimum of the divergence between the model family and the family of variational approximations. However, except in degenerate cases, this model cannot be at a stationary point in likelihood. These degenerate cases occur when the model satisfies the simplifying conditions that define the family of variational approximations. In this case, the variational EM algorithm is essentially performing standard EM over a restricted model family defined so as to be consistent with the variational approximations.

6. Conclusion

GAM iterative estimation procedures are a class of EM extensions whose E step can be varied in a manner analogous to the relaxation of the M step that occurs in GEM algorithms. We have provided conditions under which these procedures can be shown to converge to stationary points in likelihood. The conditions specify allowable E step variations that are in fact analogous to the M step variations that are allowed by GEM procedures. The convergence analysis is analogous to that presented by Wu (1983), but takes advantage of the information geometric framework of Csiszár and Tusnády (1984) to explicitly represent distributions in computing sufficient statistics.

We have analyzed the convergence behavior of two well known EM extensions, namely incremental EM and variational EM, as GAM procedures. Our GAM convergence analysis shows that incremental EM procedures converge to stationary points in likelihood, even though incremental EM is in general neither a GEM procedure nor monotonic in likelihood. Variational EM algorithms with unique E steps satisfy the conditions of the GAM convergence theorem but do not satisfy its corollary. Thus the GAM convergence theorem shows that such algorithms converge to solutions that minimize divergence, but these are not necessarily stationary points in likelihood. We then present an information geometric argument which shows that variational EM can only converge to stationary points in likelihood in degenerate cases.

Appendix A. EM Satisfies the GAM Convergence Theorem

Recall that the forward and backward steps $F, B : \mathcal{D} \times \Theta \rightarrow \mathcal{D} \times \Theta$ of the EM algorithm are given by

$$F(Q_X, \theta) = \left\{ (Q'_X, \theta) : Q'_X \in \arg \min_{Q'_X \in \mathcal{D}} D(Q'_X || P_{X;\theta}) \right\}$$

and

$$B(Q_X, \theta) = \left\{ (Q_X, \phi) : \phi \in \arg \min_{\xi \in \Theta} D(Q_X || P_{X;\xi}) \right\}.$$

That the conditions (GAM.F), (GAM.B), (EQ.F), and (EQ.B) hold is obvious by construction. We will first show a compact \mathcal{D}' that guarantees that $F(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$ and $B(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$. We will then show that F and B are closed on $\mathcal{D}' \times \Theta$. Proposition 5 then implies that the composite map FB satisfies the conditions of the GAM convergence theorem (Theorem 3).

Restricting the desired family to a compact set We define \mathcal{D}' as

$$\mathcal{D}' = \{Q_X \in \mathcal{D} : q_{X|Y}(x|\hat{y}) = p_{X|Y}(x|\hat{y}; \theta) \text{ for some } \theta \in \Theta\}$$

with \mathcal{D} defined as in Section 2. Note that this forces every $Q_X \in \mathcal{D}'$ to be the continuous mapping of some $\theta \in \Theta$. Therefore, \mathcal{D}' is the continuous mapping of the compact set Θ , and is therefore compact.

By construction of \mathcal{D}' , it is guaranteed that Q_X generated by a forward step will lie in \mathcal{D}' . Thus, $F(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$. By definition of $B(\cdot)$, $B(\mathcal{D}' \times \Theta) \subseteq \mathcal{D}' \times \Theta$.

Closedness of the forward and backward steps The following proposition and corollary show that the minimization of a continuous function forms a closed point-to-set map. This implies that projection under the divergence forms a closed point-to-set-map, so that the (ungeneralized) forward and backward steps of the EM algorithm are in fact closed point-to-set maps.

Proposition 7 *Given a real-valued continuous function f on $A \times B$, define the point-to-set map $F : A \rightarrow B$ by*

$$\begin{aligned} F(a) &= \arg \min_{b' \in B} f(a, b'), \\ &= \{b : f(a, b) \leq f(a, b') \text{ for } \forall b' \in B\}. \end{aligned}$$

Then, the point-to-set map F is closed at a if $F(a)$ is nonempty.

Proof Let $\{a^{(t)}\}_{t=0}^\infty$ and $\{b^{(t)}\}_{t=0}^\infty$ be sequences in A and B respectively, such that

$$\begin{aligned} a^{(t)} &\rightarrow a, \\ b^{(t)} &\rightarrow b, \end{aligned}$$

and suppose

$$b^{(t)} \in F\left(a^{(t)}\right).$$

That is,

$$b^{(t)} \in \arg \min_{b' \in B} f(a^{(t)}, b').$$

The map F is closed at $a \in A$ if this implies that $b \in F(a)$ – that is, that $b \in \arg \min_{b' \in B} f(a, b')$.

To prove the proposition by contradiction, suppose $b \notin \arg \min_{b' \in B} f(a, b')$. By assumption $F(a)$ is nonempty. Therefore, there exists $\hat{b} \in \arg \min_{b' \in B} f(a, b')$. Choose $\varepsilon > 0$ such that

$$f(a, b) > f(a, \hat{b}) + 2\varepsilon. \tag{9}$$

By continuity of $f(\cdot, \cdot)$ and f -monotonicity of $(a^{(t)}, b^{(t)})$, $\exists K_1$ such that

$$f(a^{(t)}, b^{(t)}) > f(a, b) - \varepsilon, \quad \forall t > K_1,$$

so that by equation (9),

$$f(a^{(t)}, b^{(t)}) > f(a, \hat{b}) + \varepsilon, \quad \forall t > K_1.$$

By continuity of $f(\cdot, \hat{b})$ and f -monotonicity of $(a^{(t)}, b^{(t)})$, $\exists K_2$ such that

$$f(a, \hat{b}) + \varepsilon > f(a^{(t)}, \hat{b}), \quad \forall t > K_2.$$

Combining these two bounds gives $\exists t > K_1, K_2$ such that

$$f(a^{(t)}, b^{(t)}) > f(a^{(t)}, \hat{b})$$

which is a contradiction since by assumption, $b^{(t)} \in \arg \min_{b' \in B} f(a^{(t)}, b')$, and therefore,

$$b \in \arg \min_{b' \in B} f(a, b'),$$

$b \in F(a)$. ■

Corollary 8 *The point-to-set map $F : A \rightarrow B$ of Proposition 7 is closed on A if the set B is closed.*

The following lemma shows that the Cartesian product of two closed point-to-set-maps is itself closed.

Lemma 9 *Suppose $F : A \rightarrow B$ and $G : A \rightarrow C$ are closed point-to-set-maps. Then the product point-to-set-map $H : A \rightarrow B \times C$ defined by*

$$H(a) = F(a) \times G(a)$$

is closed.

This follows by direct application of the definition of closedness of point-to-set maps.

Proposition 7 and the existence of the I-projection shows that the mapping from Θ to \mathcal{D} defined by

$$Q_X \in \arg \min_{Q'_X \in \mathcal{D}} D(Q'_X || P_{X;\theta})$$

is closed. This result together with Lemma 9 then show that the forward step F of the EM algorithm shown above is closed. Similarly, it can be shown using Corollary 8 and Lemma 9 that the backward step B is closed.

Appendix B. Incremental EM: (EQ) and \mathcal{D}'

In this appendix, we show that incremental EM satisfies condition EQ of the GAM convergence theorem, and show a compact restriction of the desired family that can be used to analyze convergence of incremental EM.

B.1 Incremental EM Satisfies Condition (EQ)

$(Q'_X, \theta') \in FB(Q_X, \theta)$ implies a sequence of incremental steps

$$(R_X^{(0)}, \phi^{(0)}), \dots, (R_X^{(n)}, \phi^{(n)})$$

such that

$$\begin{aligned} (R_X^{(0)}, \phi^{(0)}) &= (Q_X, \theta), \\ (R_X^{(j)}, \phi^{(j)}) &\in F^{(j)}B(R_X^{(j-1)}, \phi^{(j-1)}), \end{aligned}$$

and

$$(Q'_X, \theta') = (R_X^{(n)}, \phi^{(n)}).$$

When $D(Q'_X || P_{X;\theta'}) = D(Q_X || P_{X;\theta})$, the GAM inequality (already shown) gives that the divergence is unchanged at every incremental step:

$$D(R_X^{(j)} || P_{X;\phi^{(j)}}) = D(R_X^{(j-1)} || P_{X;\phi^{(j-1)}})$$

for $j = 1, \dots, n$. In fact, by conditions (GAM.F) and (GAM.B), the divergence is unchanged at each incremental forward step $F^{(j)}$ and the backward step B :

$$D(R_X^{(j)} || P_{X;\phi^{(j-1)}}) = D(R_X^{(j-1)} || P_{X;\phi^{(j-1)}}) \tag{10}$$

and

$$D(R_X^{(j)} || P_{X;\phi^{(j)}}) = D(R_X^{(j)} || P_{X;\phi^{(j-1)}}) \tag{11}$$

for $j = 1, \dots, n$. We will now show that $Q_{X^{(i)}} = P_{X^{(i)}|Y^{(i)}=\hat{y}^{(i)}; \phi^{(i-1)}}$ for $i = 1, \dots, n$ and that $\phi^{(j)} = \theta$ for $j = 1, \dots, n$, which will then imply that condition (EQ) holds.

To show $Q_{X^{(i)}} = P_{X^{(i)}|Y^{(i)}=\hat{y}^{(i)}; \phi^{(i-1)}}$ we decompose equation (10) as

$$\begin{aligned} \sum_{i \neq j} D(R_{X^{(i)}}^{(j)} || P_{X;\phi^{(j-1)}}) + D(R_{X^{(j)}}^{(j)} || P_{X;\phi^{(j-1)}}) = \\ \sum_{i \neq j} D(R_{X^{(i)}}^{(j-1)} || P_{X;\phi^{(j-1)}}) + D(R_{X^{(j)}}^{(j-1)} || P_{X;\phi^{(j-1)}}). \end{aligned}$$

Since $R_{X^{(i)}}^{(j)} = R_{X^{(i)}}^{(j-1)}$ for all $i \neq j$ at any incremental step (j) , this reduces to

$$D(R_{X^{(j)}}^{(j)} || P_{X;\phi^{(j-1)}}) = D(R_{X^{(j)}}^{(j-1)} || P_{X;\phi^{(j-1)}})$$

for $j = 1, \dots, n$. Since $R_{X^{(j)}}^{(j)} = P_{X^{(j)}|Y^{(j)}=\hat{y}^{(j)}; \phi^{(j-1)}}$ uniquely minimizes the component divergence $D(R_{X^{(j)}} || P_{X; \phi^{(j-1)}})$ over all $R_{X^{(j)}}$, this means that

$$R_{X^{(j)}}^{(j-1)} = R_{X^{(j)}}^{(j)} = P_{X^{(j)}|Y^{(j)}=\hat{y}^{(j)}; \phi^{(j-1)}}.$$

Thus, for any component (i) , substituting in $j = i$ and recalling that the first $i - 1$ incremental E steps leave the component marginal $R_{X^{(i)}}$ unchanged, we get

$$Q_{X^{(i)}} = R_{X^{(i)}}^{(0)} = \dots = R_{X^{(i)}}^{(i)} = P_{X^{(i)}|Y^{(i)}=\hat{y}^{(i)}; \phi^{(i-1)}}. \quad (12)$$

We now show that $\phi^{(j)} = \theta$ for $j = 1, \dots, n$. Since equation (11) tells us that the divergence is unchanged at any backward step (j) , both $P_{X; \phi^{(j)}}$ and $P_{X; \phi^{(j-1)}}$ must minimize $D(R_X^{(j)} || \cdot)$. By assumption (8), the M-step is uniquely determined, so we must have $\phi^{(j)} = \phi^{(j-1)}$. We therefore have the desired result

$$\theta = \phi^{(0)} = \dots = \phi^{(n)} = \theta'.$$

Substituting this into equation (12) gives

$$Q_{X^{(i)}} = R_{X^{(i)}}^{(0)} = \dots = R_{X^{(i)}}^{(i)} = P_{X^{(i)}|Y^{(i)}=\hat{y}^{(i)}; \theta}.$$

Since this applies for all $i = 1, \dots, n$, we have

$$R_X^{(j)} = P_{X|Y=\hat{y}; \theta}, \quad \forall j = 0, \dots, n.$$

In particular, $Q_X = R_X^{(0)} = P_{X|Y=\hat{y}; \theta}$, which means that

$$Q_X = \operatorname{argmin}_{Q_X'' \in \mathcal{D}} D(Q_X'' || P_{X; \theta}).$$

Since $R_X^{(1)} = Q_X$, we use equation (11) with $j = 1$ and condition (EQ.B) on the backward map to get

$$\theta \in \operatorname{argmin}_{\xi \in \Theta} D(Q_X || P_{X; \xi}).$$

This shows that condition (EQ) holds.

B.2 Definition of a Compact \mathcal{D}'

To find a suitable restriction \mathcal{D}' for any choice of $Q_X^{(0)} \in \mathcal{D}$, we first define the following sets of measures on the components $X^{(i)}$:

$$\mathcal{D}_{INC}^{(i)} = \left\{ Q_{X^{(i)}} : Q_{X^{(i)}} = P_{X^{(i)}|Y^{(i)}=\hat{y}^{(i)}; \theta} \text{ for some } \theta \in \Theta \right\} \cup \left\{ Q_{X^{(i)}}^{(0)} \right\},$$

and note that the continuity of $P_{X|Y; \theta}$ (assumed), and the compactness of Θ (assumed) give us compactness of $\mathcal{D}_{INC}^{(i)}$. We then define our restriction \mathcal{D}'_{INC} of \mathcal{D} by

$$\mathcal{D}'_{INC} = \left\{ Q_X : Q_X = \prod_{i=1}^n Q_{X^{(i)}} \text{ for some } (Q_{X^{(1)}}, \dots, Q_{X^{(n)}}) \in \mathcal{D}_{INC}^{(1)} \times \dots \times \mathcal{D}_{INC}^{(n)} \right\}.$$

To show compactness of \mathcal{D}'_{INC} , suppose $\{Q_X^{(t)}\}_{t=0}^{\infty}$ is a sequence in \mathcal{D}'_{INC} . From the definition of \mathcal{D}'_{INC} , this then implies that there are n sequences $\{Q_{X^{(i)}}^{(t)}\}_{t=0}^{\infty}$, each in the corresponding $\mathcal{D}_{INC}^{(i)}$, such that $Q_X^{(t)} = \prod_{i=1}^n Q_{X^{(i)}}^{(t)}$. The compactness of $\mathcal{D}_{INC}^{(1)}$ implies the existence of an infinite subset $\mathcal{K}^{(1)}$ of the integers such that the subsequence $\{Q_{X^{(1)}}^{(t)}\}_{t \in \mathcal{K}^{(1)}}$ converges to some $Q_{X^{(1)}}^* \in \mathcal{D}_{INC}^{(1)}$. Similarly, since the infinite sequence $\{Q_{X^{(i)}}^{(t)}\}_{t \in \mathcal{K}^{(i-1)}}$ is contained in the compact set $\mathcal{D}_{INC}^{(i)}$, there exists an infinite subset $\mathcal{K}^{(i)}$ of $\mathcal{K}^{(i-1)}$ such that the subsequence $\{Q_{X^{(i)}}^{(t)}\}_{t \in \mathcal{K}^{(i)}}$ converges to some $Q_{X^{(i)}}^* \in \mathcal{D}_{INC}^{(i)}$. Therefore, the subsequence $\{Q_X^{(t)}\}_{t \in \mathcal{K}^{(n)}}$ converges to $\prod_{i=1}^n Q_{X^{(i)}}^* \in \mathcal{D}'_{INC}$, showing that \mathcal{D}'_{INC} is compact.

References

- S.-I. Amari. Information geometry of the EM and *em* algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995.
- R. A. Boyles. On the convergence of the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 45(1):47–50, 1983.
- W. Byrne. Alternating minimization and Boltzman machine learning. *IEEE Transactions on Neural Networks*, 3(4):612–620, 1992.
- W. Byrne and A. Gunawardana. Comments on ‘Efficient training algorithms for HMM’s using incremental estimation’. *IEEE Transactions on Speech and Audio Processing*, 8(6):751–754, November 2000.
- I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *Annals of Probability*, 3(1):146–158, 1975.
- I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics and Decisions, Supplemental Issue Number 1*, pages 205–237, 1984.
- I. Csiszár. Information theory and statistics. ENEE 728F. Lecture Notes, Spring 1990. University of Maryland.
- J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.
- A. P. Dempster, A. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
- V. Digalakis. On-line adaptation of Hidden Markov Models using incremental estimation models. In *European Conference on Speech Communication and Technology*, pages 1859–1862, 1997.
- B. Efron. Defining the curvature of a statistical problem (with applications to second order efficiency). *Annals of Statistics*, 3(6):1189–1242, 1975.

- R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society A*, 222:309–768, 1922.
- A. Gunawardana. *The Information Geometry of EM Variants for Speech and Image Processing*. PhD thesis, The Johns Hopkins University, 2001.
- I.-T. Hsiao, A. Rangarajan, P. Khurd, and G. Gindi. Fast, globally convergent, reconstruction in emission tomography using COSEM, an incremental EM algorithm. *IEEE Transactions on Medical Imaging*, 2004. Submitted.
- M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 105–162. MIT Press, 1999.
- E. L. Lehmann. Efficient likelihood estimators. *The American Statistician*, 34(4):233–235, November 1980.
- F. Liese and I. Vajda. *Convex Statistical Distances*. Teubner Verlagsgesellschaft, Leipzig, 1987.
- G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.
- I. Meilijson. A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society, Series B*, 51(1):127–138, 1989.
- R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Press, 1998.
- R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani. On the convergence of bound optimization algorithms. In *Conference on Uncertainty in Artificial Intelligence*, volume 19, 2003.
- B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, pages 279–299, 2001.
- A. Wald. Note on the consistency of the maximum likelihood estimate. *Annals of Mathematical Statistics*, 20, 1949.
- C. F. J. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983.
- W. I. Zangwill. *Nonlinear Programming: A Unified Approach*. Prentice-Hall, 1969.

Kernel Methods for Measuring Independence

Arthur Gretton

*MPI for Biological Cybernetics
Spemannstrasse 38
72076, Tübingen, Germany*

ARTHUR@TUEBINGEN.MPG.DE

Ralf Herbrich

*Microsoft Research Cambridge
7 J. J. Thomson Avenue
Cambridge CB3 0FB, United Kingdom*

RHERB@MICROSOFT.COM

Alexander Smola

*National ICT Australia
Canberra, ACT 0200, Australia*

ALEX.SMOLA@NICTA.COM.AU

Olivier Bousquet

*Pertinence
32, Rue des Jeûneurs
75002 Paris, France*

OLIVIER.BOUSQUET@PERTINENCE.COM

Bernhard Schölkopf

*MPI for Biological Cybernetics
Spemannstrasse 38
72076, Tübingen, Germany*

BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE

Editor: Aapo Hyvärinen

Abstract

We introduce two new functionals, the constrained covariance and the kernel mutual information, to measure the degree of independence of random variables. These quantities are both based on the covariance between functions of the random variables in reproducing kernel Hilbert spaces (RKHSs). We prove that when the RKHSs are universal, both functionals are zero if and only if the random variables are pairwise independent. We also show that the kernel mutual information is an upper bound near independence on the Parzen window estimate of the mutual information. Analogous results apply for two correlation-based dependence functionals introduced earlier: we show the kernel canonical correlation and the kernel generalised variance to be independence measures for universal kernels, and prove the latter to be an upper bound on the mutual information near independence. The performance of the kernel dependence functionals in measuring independence is verified in the context of independent component analysis.

Keywords: independence, covariance operator, mutual information, kernel, Parzen window estimate, independent component analysis

1. Introduction

Measures to determine the dependence or independence of random variables are well established in statistical analysis. For instance, one well known measure of statistical dependence between two random variables is the *mutual information* (Cover and Thomas, 1991), which for random vectors \mathbf{x}, \mathbf{y} is zero if and only if the random vectors are independent. This may also be interpreted as the KL divergence $D_{\text{KL}}(\mathbf{p}_{\mathbf{x}, \mathbf{y}} \parallel \mathbf{p}_{\mathbf{x}} \mathbf{p}_{\mathbf{y}})$ between the joint density and the product of the marginal densities; the latter quantity generalises readily to distributions of more than two random variables (there exist other methods for independence measurement: see for instance Ingster, 1989).

There has recently been considerable interest in using criteria based on functions in reproducing kernel Hilbert spaces to measure dependence, notably in the context of independent component analysis.¹ This was first accomplished by Bach and Jordan (2002a), who introduced kernel dependence functionals that significantly outperformed alternative approaches, including for source distributions that are difficult for standard ICA methods to deal with. In the present study, we build on this work with the introduction of two novel kernel-based independence measures. The first, which we call the constrained covariance (COCO), is simply the spectral norm of the covariance operator between reproducing kernel Hilbert spaces. We prove COCO to be zero if and only if the random variables being tested are independent, as long as the RKHSs used to compute it are universal. The second functional, called the kernel mutual information (KMI), is a more sophisticated measure of dependence, being a function of the entire spectrum of the covariance operator. We show that the KMI is an upper bound near independence on a Parzen window estimate of the mutual information, which becomes tight (*i.e.*, zero) when the random variables are independent, again assuming universal RKHSs. Note that Gretton et al. (2003a,b) attempted to show a link with the Parzen window estimate, although this earlier proof is wrong - the reader may compare Section 3 in the present document with the corresponding section of the original technical report, since the differences are fairly obvious.²

The constrained covariance has substantial precedent in the dependence testing literature. Indeed, Rényi (1959) suggested using the functional covariance or correlation to measure the dependence of random variables (implementation details depend on the nature of the function spaces chosen: the use of RKHSs is a more recent innovation). Thus, rather than using the covariance, we may consider a kernelised canonical correlation (KCC) (Bach and Jordan, 2002a; Leurgans et al., 1993), which is a regularised estimate of the spectral norm of the *correlation* operator between reproducing kernel Hilbert spaces. It follows from the properties of COCO that the KCC is zero at independence for universal kernels, since the correlation differs from the covariance only in its normalisation: at independence, where both the KCC and COCO are zero, this normalisation is immaterial. The introduction of a regulariser requires a new parameter that must be tuned, however, which was not needed for COCO or the KMI.

Another kernel method for dependence measurement, the kernel generalised variance (KGV) (Bach and Jordan, 2002a), extends the KCC by incorporating the entire spectrum of its associated

-
1. The problem of instantaneous independent component analysis involves the recovery of linearly mixed, i.i.d. sources, in the absence of information about the source distributions beyond their mutual independence (Hyvärinen et al., 2001).
 2. Briefly, we now use Lemma 27 as a basis for our proof, which applies to every singular value of a matrix product; our earlier proof relied on Theorem 4.2.2 of Gretton et al. (2003a), which implies a result only for the largest singular value, and is therefore insufficient. On the other hand, we believe that the proof given by Gretton (2003) in Chapter 9 is correct, but the approach is a bit clumsy, and much longer than it needs to be.

	Covariance	Correlation
Max. singular value	COCO (Gretton et al., 2005b)	KCC (Bach and Jordan, 2002a)
MI bound	KMI	KGV (Bach and Jordan, 2002a)

Table 1: Table of kernel dependence functionals. Columns show whether the functional is covariance or correlation based, and rows indicate whether the dependence measure is the maximum singular value of the covariance/correlation operator, or a bound on the mutual information.

correlation operator: in this respect, the KGV and KMI are analogous (see Table 1). Indeed, we prove here that under certain reasonable and easily enforced conditions, the KGV is an upper bound on the KMI (and hence on the mutual information near independence), which also becomes tight at independence. A relation between the KGV and the mutual information is also proposed by Bach and Jordan (2002a), who rely on a limiting argument in which the RKHS kernel size approaches zero (no Parzen window estimate is invoked): our discussion of this proof is given in Appendix B.2.

We should warn the reader that results presented in this study have a conceptual emphasis: we attempt to build on the work of Bach and Jordan (2002a) by on one hand exploring the mechanism by which kernel covariance operator-based functionals measure independence (including a characterisation of all kernels that induce independence measures), and on the other hand demonstrating the link between kernel dependence functionals and the mutual information. That said, we observe differences in practice when the various kernel methods are applied in ICA: the KMI generally outperforms the KGV for many sources/large sample sizes, whereas the KGV gives best performance for small sample sizes. The choice of regulariser for the KGV (and KCC) is also crucial, since a badly chosen regularisation is severely detrimental to performance when outlier noise is present. The KMI and COCO are robust to outliers, and yield experimental performance equivalent to the KGV and KCC with optimal regulariser choice, but without any tuning required.

The COCO and KCC dependence functionals for the 2-variable case are described in Section 2, and it is shown that these measure independence when the associated kernels are universal. The main results in this section are Definition 2, which presents both the population COCO and its empirical counterpart, and Theorem 6, which shows that COCO is an independence measure. Section 3 contains derivations of the kernel-based upper bounds on the mutual information, and proofs that these latter quantities likewise measure independence. In particular, the kernel mutual information is introduced in Definition 14, its use as an independence measure is justified by Theorem 15, and its relation to the mutual information is provided in Theorem 16. A generalisation to more than two variables, which permits the measurement of pairwise independence, is also presented. Section 4 addresses the application of kernel dependence measures to independent component analysis, including a method for reducing computational cost and a gradient descent technique (these being adapted straightforwardly from Bach and Jordan, 2002a). Finally, Section 5 describes our experiments: these demonstrate that the performance of the KMI and COCO, when used in ICA, is competitive with the KGV and KCC, respectively. The kernel methods also compare favourably with both standard and recent specialised ICA algorithms (RADICAL, CFICA, Fast ICA, Jade, and

Acronym	Description
COCO	Constrained covariance
ICA	Independent component analysis
KCC	Kernel canonical correlation
KGV	Kernel generalised variance
KMI	Kernel mutual information
RKHS	Reproducing kernel Hilbert space

Table 2: Table of acronyms

Infomax), and outperform these methods when demixing music sources (where the sample size is large). Most interestingly, when the KGV is made to approach the KMI by an appropriate choice of regularisation, its resistance to outlier noise is improved — moreover, kernel methods perform substantially better than the other algorithms tested when outliers are present.³ We list our most commonly used acronyms in Table 2.

2. Constrained Covariance, Kernel Canonical Correlation

In this section, we focus on the formulation of measures of independence for two random variables. This reasoning uses well established principles, going back to Rényi (1959), who gave a list of desirable properties for a measure of statistical dependence $\mathcal{Q}(\mathbf{P}_{x,y})$ between random variables x, y with distribution $\mathbf{P}_{x,y}$. These include

1. $\mathcal{Q}(\mathbf{P}_{x,y})$ is well defined,
2. $0 \leq \mathcal{Q}(\mathbf{P}_{x,y}) \leq 1$,
3. $\mathcal{Q}(\mathbf{P}_{x,y}) = 0$ if and only if x, y independent,
4. $\mathcal{Q}(\mathbf{P}_{x,y}) = 1$ if and only if $y = f(x)$ or $x = g(y)$, where f and g are Borel measurable functions.

Rényi (1959) shows that one measure satisfying these constraints is

$$\mathcal{Q}(\mathbf{P}_{x,y}) = \sup_{f,g} \text{corr}(f(x), g(y)),$$

where $f(x), g(y)$ must have finite positive variance, and f, g are Borel measurable. This is similar to the kernel canonical correlation (KCC) introduced by Bach and Jordan (2002a), although we shall see that the latter is more restrictive in its choice of f, g . We propose a different measure, the *constrained covariance* (COCO), which omits the fourth property and the upper bound in the second property; in the context of independence measurement, however, the first and third properties are adequate.⁴

3. The performance reported here improves on that obtained by Bach and Jordan (2002a); Learned-Miller and Fisher III (2003) due to better tuning of the KGV and KCC regularisation.

4. The fourth property is required for \mathcal{Q} to identify deterministic dependence, which an *independence* measure should not be concerned with.

We begin in Section 2.1 by defining RKHSs and covariance operators between them. In Section 2.2, we introduce the constrained covariance, and we demonstrate in Section 2.3 that this quantity is a measure of independence when computed in universal RKHSs (it follows that the KCC also requires a universal RKHS, as do all independence criteria that are based on the covariance in RKHSs). Finally, we describe the canonical correlation in Section 2.4, and its RKHS-based variant.

2.1 Covariance in Function Spaces

In this section, we provide the functional analytic background necessary in describing covariance operators between RKHSs. Our presentation follows and extends the work of Zwald et al. (2004); Hein and Bousquet (2004), who deal with covariance operators from a space to itself rather than from one space to another, and Fukumizu et al. (2004), who use covariance operators as a means of defining conditional covariance operators. Functional covariance operators were investigated earlier by Baker (1973), who characterises these operators for general Hilbert spaces.

Consider a Hilbert space \mathcal{F} of functions from \mathcal{X} to \mathbb{R} , where \mathcal{X} is a separable metric space. The Hilbert space \mathcal{F} is an RKHS if at each $x \in \mathcal{X}$, the point evaluation operator $\delta_x : \mathcal{F} \rightarrow \mathbb{R}$, which maps $f \in \mathcal{F}$ to $f(x) \in \mathbb{R}$, is a bounded linear functional. To each point $x \in \mathcal{X}$, there corresponds an element $\mathbf{x} := \phi(x) \in \mathcal{F}$ (we call ϕ the *feature map*) such that $\langle \phi(x), \phi(x') \rangle_{\mathcal{F}} = k(x, x')$, where $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a unique positive definite kernel. We also define a second RKHS \mathcal{G} with respect to the separable metric space \mathcal{Y} , with feature map ψ and kernel $\langle \psi(y), \psi(y') \rangle_{\mathcal{G}} = l(y, y')$.

Let $\mathbf{P}_{x,y}(x, y)$ be a joint measure⁵ on $(\mathcal{X} \times \mathcal{Y}, \Gamma \times \Lambda)$ (here Γ and Λ are the Borel σ -algebras on \mathcal{X} and \mathcal{Y} , respectively, as required in Theorem 4 below), with associated marginal measures \mathbf{P}_x and \mathbf{P}_y and random variables x and y . Then following Baker (1973); Fukumizu et al. (2004), the covariance operator $C_{xy} : \mathcal{G} \rightarrow \mathcal{F}$ is defined⁶ such that for all $f \in \mathcal{F}$ and $g \in \mathcal{G}$,

$$\langle f, C_{xy}g \rangle_{\mathcal{F}} = \mathbf{E}_{x,y} ([f(x) - \mathbf{E}_x(f(x))] [g(y) - \mathbf{E}_y(g(y))]).$$

In practice, we do not deal with the measure $\mathbf{P}_{x,y}$ itself, but instead observe samples drawn independently according to it. We write an i.i.d. sample of size m from $\mathbf{P}_{x,y}$ as $\mathbf{z} = \{(x_1, y_1), \dots, (x_m, y_m)\}$, and likewise $\mathbf{x} := \{x_1, \dots, x_m\}$ and $\mathbf{y} := \{y_1, \dots, y_m\}$. Finally, we define the Gram matrices \mathbf{K} and \mathbf{L} of inner products in \mathcal{F} and \mathcal{G} , respectively, between the mapped observations above: here \mathbf{K} has (i, j) th entry $k(x_i, x_j)$ and \mathbf{L} has (i, j) th entry $l(y_i, y_j)$. The Gram matrices for the variables centred in their respective feature spaces are shown by Schölkopf et al. (1998) to be

$$\tilde{\mathbf{K}} := \mathbf{H}\mathbf{K}\mathbf{H}, \quad \tilde{\mathbf{L}} := \mathbf{H}\mathbf{L}\mathbf{H},$$

where

$$\mathbf{H} = \mathbf{I} - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m^\top, \tag{1}$$

and $\mathbf{1}_m$ is an $m \times 1$ vector of ones.

5. We do not require this to have a density with respect to a reference measure $dx \times dy$ in this section. Note that we will need a density in Section 3, however.

6. Our operator (and that of Fukumizu et al., 2004) differs from Baker's in that Baker defines all measures directly on the function spaces.

2.2 The Constrained Covariance

In this section, we define the constrained covariance (COCO), and describe the properties of the kernelised version. The covariance between \mathbf{x} and \mathbf{y} is defined as follows.

Definition 1 (Covariance) *The covariance of two random variables \mathbf{x}, \mathbf{y} is given as*

$$\text{cov}(\mathbf{x}, \mathbf{y}) := \mathbf{E}_{\mathbf{x}, \mathbf{y}}[\mathbf{x}\mathbf{y}] - \mathbf{E}_{\mathbf{x}}[\mathbf{x}]\mathbf{E}_{\mathbf{y}}[\mathbf{y}].$$

We next define the constrained covariance.

Definition 2 (Constrained Covariance (COCO)) *Given function classes \mathcal{F}, \mathcal{G} and a probability measure $\mathbf{P}_{\mathbf{x}, \mathbf{y}}$, we define the constrained covariance as*

$$\text{COCO}(\mathbf{P}_{\mathbf{x}, \mathbf{y}}; \mathcal{F}, \mathcal{G}) := \sup_{f \in \mathcal{F}, g \in \mathcal{G}} [\text{cov}(f(\mathbf{x}), g(\mathbf{y}))]. \quad (2)$$

If \mathcal{F} and \mathcal{G} are unit balls in their respective vector spaces, then this is just the norm of the covariance operator: see Mourier (1953). Given m independent observations $\mathbf{z} := ((x_1, y_1), \dots, (x_m, y_m)) \subset (\mathcal{X} \times \mathcal{Y})^m$, the empirical estimate of COCO is defined as

$$\text{COCO}(\mathbf{z}; \mathcal{F}, \mathcal{G}) := \sup_{f \in \mathcal{F}, g \in \mathcal{G}} \left[\frac{1}{m} \sum_{i=1}^m f(x_i)g(y_i) - \frac{1}{m^2} \sum_{i=1}^m f(x_i) \sum_{j=1}^m g(y_j) \right].$$

When \mathcal{F} and \mathcal{G} are RKHSs, with F and G their respective unit balls, then $\text{COCO}(\mathbf{P}_{\mathbf{x}, \mathbf{y}}; F, G)$ is guaranteed to exist as long as the kernels k and l are bounded, since the covariance operator is then Hilbert-Schmidt (as shown by Gretton et al., 2005a). The empirical estimate $\text{COCO}(\mathbf{z}; F, G)$ is also simplified when F and G are unit balls in RKHSs, since the representer theorem (Schölkopf and Smola, 2002) holds: this states that a solution of an optimisation problem, dependent only on the function evaluations on a set of observations and on RKHS norms, lies in the span of the kernel functions evaluated on the observations. This leads to the following lemma:

Lemma 3 (Value of $\text{COCO}(\mathbf{z}; F, G)$) *Denote by \mathcal{F} and \mathcal{G} RKHSs on the domains X and \mathcal{Y} respectively, and let F, G be the unit balls in the corresponding RKHSs. Then*

$$\text{COCO}(\mathbf{z}; F, G) = \frac{1}{m} \sqrt{\|\tilde{\mathbf{K}}\tilde{\mathbf{L}}\|_2}, \quad (3)$$

where the matrix norm $\|\cdot\|_2$ denotes the largest singular value. An equivalent unnormalised form (which we will refer back to in Section 3) is $\text{COCO}(\mathbf{z}; F, G) = \max_i \gamma_i$, where γ_i are the solutions to the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ \tilde{\mathbf{L}}\tilde{\mathbf{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_i \\ \boldsymbol{\beta}_i \end{bmatrix} = \gamma_i \begin{bmatrix} \tilde{\mathbf{K}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{L}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_i \\ \boldsymbol{\beta}_i \end{bmatrix}. \quad (4)$$

Proof By the representer theorem, the solution of the maximisation problem arising from $\text{COCO}(\mathbf{z}; F, G)$ is given by $f(x) = \sum_{i=1}^m \alpha_i k(x_i, x)$ and $g(y) = \sum_{j=1}^m \beta_j l(y_j, y)$. Hence

$$\begin{aligned} \text{COCO}(\mathbf{z}; F, G) &= \sup_{\boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \leq 1, \boldsymbol{\beta}^\top \mathbf{L} \boldsymbol{\beta} \leq 1} \frac{1}{m} \boldsymbol{\alpha}^\top \mathbf{K} \mathbf{L} \boldsymbol{\beta} - \frac{1}{m^2} \boldsymbol{\alpha}^\top \mathbf{K} \mathbf{1}_m \mathbf{1}_m^\top \mathbf{L} \boldsymbol{\beta} \\ &= \sup_{\|\boldsymbol{\alpha}\|, \|\boldsymbol{\beta}\| \leq 1} \frac{1}{m} \boldsymbol{\alpha}^\top \mathbf{K}^{1/2} \mathbf{H} \mathbf{L}^{1/2} \boldsymbol{\beta} \\ &= \frac{1}{m} \|\mathbf{K}^{1/2} \mathbf{H} \mathbf{L}^{1/2}\|_2. \end{aligned}$$

Squaring the argument in the norm, rearranging, and using the fact that $\mathbf{H} = \mathbf{H} \mathbf{H}$ proves the lemma. ■

The constrained covariance turns out to be similar in certain respects to a number of kernel algorithms, for an appropriate choice of \mathcal{F}, \mathcal{G} . By contrast with independence measurement, however, these methods seek to *maximise* the constrained covariance through the correct choice of feature space elements. First, and most obvious, is kernel partial least squares (kPLS) (Rosipal and Trejo, 2001), which at each stage maximises the constrained covariance directly (see Bakır et al., 2004). COCO is also optimised when obtaining the first principal component in kernel principal component analysis (kPCA), as described by Schölkopf et al. (1998), and is the criterion optimised in the spectral clustering/kernel target alignment framework of Cristianini et al. (2002). Details may be found in Appendix A.1.

Finally, we remark that alternative norms of the covariance operator should also be suited to measuring independence. Indeed, the Hilbert-Schmidt (HS) norm is proposed in this context by Gretton et al. (2005a): like the KMI, it exploits the entire spectrum of the empirical covariance operator, and gives experimental performance superior to COCO in ICA. The HS norm has the additional advantage of a well-defined population counterpart, and guarantees of $O(1/\sqrt{m})$ convergence of the empirical to the population quantity. The connection between the HS norm and the mutual information remains unknown, however.

2.3 Independence Measurement with the Constrained Covariance

We now describe how COCO is used as a measure of independence. For our purposes, the notion of independence of random variables is best characterised by Jacod and Protter (2000, Theorem 10.1(e)):

Theorem 4 (Independence) *Let x and y be random variables on $(\mathcal{X} \times \mathcal{Y}, \Gamma \times \Lambda)$ with joint measure $\mathbf{P}_{x,y}(x, y)$, where Γ and Λ are Borel σ -algebras on \mathcal{X} and \mathcal{Y} , respectively. Then the random variables x and y are independent if and only if $\text{cov}(f(x), g(y)) = 0$ for any pair (f, g) of bounded, continuous functions.*

It follows from Theorem 4 that if \mathcal{F}, \mathcal{G} are the sets of bounded continuous functions, then $\text{COCO}(\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G}) = 0$ if and only if x and y are independent. In other words, $\text{COCO}(\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G})$ and $\text{COCO}(\mathbf{z}; \mathcal{F}, \mathcal{G})$ are criteria which can be tested *directly* without the need for an intermediate density estimator (in general, the distributions may not even have densities). It is also clear, however, that unless \mathcal{F}, \mathcal{G} are restricted in further ways, $\text{COCO}(\mathbf{z}; \mathcal{F}, \mathcal{G})$ will always be large, due to the rich choice of functions available. A *non-trivial dependence functional* is thus obtained using function

classes that do not give an everywhere-zero empirical average, yet which still guarantee that COCO is zero if and only if its arguments are independent. A tradeoff between the restrictiveness of the function classes and the convergence of $\text{COCO}(\mathbf{z}; \mathcal{F}, \mathcal{G})$ to $\text{COCO}(\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G})$ can be accomplished using standard tools from uniform convergence theory (see Gretton et al., 2005b). It turns out that unit-radius balls in *universal* reproducing kernel Hilbert spaces constitute function classes that yield non-trivial dependence estimates. Universality is defined by Steinwart (2001) as follows:

Definition 5 (Universal kernel) A continuous kernel $k(\cdot, \cdot)$ on a compact metric space (X, d) is called *universal* if and only if the RKHS \mathcal{F} induced by the kernel is dense in $C(X)$, the space of continuous functions on X , with respect to the infinity norm $\|f - g\|_\infty$.

Steinwart (2001) shows the following two kernels are universal on compact subsets of \mathbb{R}^d :

$$\begin{aligned} k(x, x') &= \exp(-\lambda \|x - x'\|^2) \text{ and} \\ k(x, x') &= \exp(-\lambda \|x - x'\|) \text{ for } \lambda > 0. \end{aligned}$$

We now state our main result for this section.

Theorem 6 (COCO($\mathbf{P}_{x,y}; F, G$) is only zero at independence for universal kernels) Denote by \mathcal{F} and \mathcal{G} RKHSs with universal kernels on the compact metric spaces X and \mathcal{Y} , respectively, and let F, G be the unit balls in \mathcal{F} and \mathcal{G} . Then $\text{COCO}(\mathbf{P}_{x,y}; F, G) = 0$ if and only if x, y are independent.

Proof It is clear that $\text{COCO}(\mathbf{P}_{x,y}; F, G)$ is zero if x and y are independent. We prove the converse by showing that⁷ $\text{COCO}(\mathbf{P}_{x,y}; B(X), B(\mathcal{Y})) = c$ for some $c > 0$ implies $\text{COCO}(\mathbf{P}_{x,y}; F, G) = d$ for $d > 0$: this is equivalent to $\text{COCO}(\mathbf{P}_{x,y}; F, G) = 0$ implying $\text{COCO}(\mathbf{P}_{x,y}; B(X), B(\mathcal{Y})) = 0$ (where this last result implies independence by Theorem 4). There exist two sequences of functions $f_n \in C(X)$ and $g_n \in C(\mathcal{Y})$, satisfying $\|f_n\|_\infty \leq 1, \|g_n\|_\infty \leq 1$, for which

$$\lim_{n \rightarrow \infty} \text{cov}(f_n(x), g_n(y)) = c.$$

More to the point, there exists an n^* for which $\text{cov}(f_{n^*}(x), g_{n^*}(y)) \geq c/2$. We know that \mathcal{F} and \mathcal{G} are respectively dense in $C(X)$ and $C(\mathcal{Y})$ with respect to the L_∞ norm: this means that for all $\frac{c}{24} > \varepsilon > 0$, we can find some $f^* \in \mathcal{F}$ (and an analogous $g^* \in \mathcal{G}$) satisfying $\|f^* - f_{n^*}\|_\infty < \varepsilon$. Thus, we obtain

$$\begin{aligned} \text{cov}(f^*(x), g^*(y)) &= \text{cov}(f^*(x) - f_{n^*}(x) + f_{n^*}(x), g^*(x) - g_{n^*}(x) + g_{n^*}(x)) \\ &= \mathbf{E}_{x,y} [(f^*(x) - f_{n^*}(x) + f_{n^*}(x)) (g^*(y) - g_{n^*}(y) + g_{n^*}(y))] \\ &\quad - \mathbf{E}_x (f^*(x) - f_{n^*}(x) + f_{n^*}(x)) \mathbf{E}_y (g^*(y) - g_{n^*}(y) + g_{n^*}(y)) \\ &\geq \text{cov}(f_{n^*}(x), g_{n^*}(y)) - 2\varepsilon |\mathbf{E}_x (f_{n^*}(x))| - 2\varepsilon |\mathbf{E}_y (g_{n^*}(y))| - 2\varepsilon^2 \\ &\geq \frac{c}{2} - 6\frac{c}{24} = \frac{c}{4} > 0. \end{aligned}$$

Finally, bearing in mind that $\|f^*(x)\|_{\mathcal{F}} < \infty$ and $\|g^*(x)\|_{\mathcal{G}} < \infty$, we have

$$\text{cov} \left(\frac{f^*(x)}{\|f^*(x)\|_{\mathcal{F}}}, \frac{g^*(y)}{\|g^*(x)\|_{\mathcal{G}}} \right) \geq \frac{c}{4 \|f^*(x)\|_{\mathcal{F}} \|g^*(x)\|_{\mathcal{G}}} > 0,$$

7. Here $B(X)$ denotes the subset of $C(X)$ of continuous functions bounded by 1 in $L_\infty(X)$, and $B(\mathcal{Y})$ is defined in an analogous manner.

and hence $\text{COCO}(\mathbf{P}_{x,y}; F, G) > 0$. ■

The constrained covariance is further explored by Gretton et al. (2005b, 2004). We prove two main results in these studies, which are not covered in the present work:

- Theorems 10 and 11 of Gretton et al. (2005b) give upper bounds on the probability of large deviations of the empirical COCO from the population COCO: Theorem 10 covers negative deviations of the empirical COCO from the population COCO, and Theorem 11 describes positive deviations. For a fixed probability of deviation, the amount by which the empirical COCO differs from the population COCO decreases at rate $1/\sqrt{m}$ (for shifts in either direction). These bounds are necessary if we are to formulate *statistical tests* of independence based on the *measure* of independence that COCO provides. In particular, Gretton et al. (2005b, Section 5) give one such test .
- Theorem 8 of Gretton et al. (2005b) describes the behaviour of the population COCO when the random variables are not independent, for a simple family of probability densities represented as orthogonal series expansions. This is used to illustrate two concepts: first, that dependence can sometimes be hard to detect without a large number of samples (since the deviation of the population COCO from zero can be very small, even for dependent random variables); and second, that one type of hard-to-detect dependence is encoded in high frequencies of the probability density function.

We also apply COCO in these studies to detecting dependence in fMRI scans of the Macaque visual cortex. We refer the reader to these references for further detail on COCO.

2.4 The Canonical Correlation

The kernelised canonical correlation (KCC) — i.e., the norm of the *correlation operator* between RKHSs — was proposed as a measure of independence by Bach and Jordan (2002a). Consistency of the KCC was shown by Leurgans et al. (1993) for the operator norm, and by Fukumizu et al. (2005) for the functions in \mathcal{F} and \mathcal{G} that define it (in accordance with Definition 7 below). Further discussion and applications of the kernel canonical correlation include Akaho (2001); Bach and Jordan (2002a); Haroon et al. (2004); Kuss (2001); Lai and Fyfe (2000); Melzer et al. (2001); Shawe-Taylor and Cristianini (2004); van Gestel et al. (2001). In particular, a much more extensive discussion of the properties of canonical correlation analysis and its kernelisation may be found in these studies, and this section simply summarises the properties and derivations relevant to our requirements for independence measurement.

The idea underlying the KCC is to find the functions $f \in \mathcal{F}$ and $g \in \mathcal{G}$ with largest *correlation* (as opposed to covariance, which we covered in the previous section). This leads to the following definition.

Definition 7 (Kernel canonical correlation (KCC)) *The kernel canonical correlation is defined as*

$$\begin{aligned} \text{KCC}(\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G}) &= \sup_{f \in \mathcal{F}, g \in \mathcal{G}} \text{corr}(f(x), g(y)) \\ &= \sup_{f \in \mathcal{F}, g \in \mathcal{G}} \frac{\mathbf{E}(f(x)g(y)) - \mathbf{E}_x(f(x))\mathbf{E}_y(g(y))}{\sqrt{\mathbf{E}_x(f^2(x)) - \mathbf{E}_x^2(f(x))}\sqrt{\mathbf{E}_y(g^2(y)) - \mathbf{E}_y^2(g(y))}}. \end{aligned}$$

As in the case of the constrained covariance, we may specify an empirical estimate similar to that in Lemma 3:

Lemma 8 (Empirical KCC) *The empirical kernel canonical correlation is given by $\text{KCC}(\mathbf{z}; \mathcal{F}, \mathcal{G}) := \max_i(\rho_i)$, where ρ_i are the solutions to the generalised eigenvalue problem*

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ \tilde{\mathbf{L}}\tilde{\mathbf{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} \tilde{\mathbf{K}}^2 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{L}}^2 \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix}. \quad (5)$$

Bach and Jordan (2002a) point out that the first canonical correlation is very similar to the function maximised by the *alternating conditional expectation* algorithm of Breiman and Friedman (1985), although in the latter case $f(x)$ may be replaced with a linear combination of several functions of x .

We note that the numerator of the functional in Definition 7 is just the functional covariance, which suggests that the kernel canonical correlation might also be a useful measure of independence: this was proposed by Bach and Jordan (2002a) (the functional correlation was also analysed as an independence measure by Dauxois and Nkiet (1998), although this approach did not make use of RKHSs). A problem with using the kernel canonical correlation to measure independence is discussed in various forms by Bach and Jordan (2002a); Fukumizu et al. (2005); Greenacre (1984); Kuss (2001); Leurgans et al. (1993); we now describe one formulation of problem, and the two main ways in which it has been solved.

Lemma 9 (Without regularisation, the empirical KCC is independent of the data) *Suppose that the Gram matrices \mathbf{K} and \mathbf{L} have full rank. The $2(m-1)$ non-zero solutions to (5) are then $\rho_i = \pm 1$, regardless of \mathbf{z} .*

The proof is in Appendix B.1. This argument is used by Bach and Jordan (2002a); Fukumizu et al. (2005); Leurgans et al. (1993) to justify a regularised canonical correlation,

$$\text{KCC}(\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G}, \kappa) := \sup_{f \in \mathcal{F}, g \in \mathcal{G}} \frac{\text{cov}(f(x), g(y))}{\left(\text{var}(f(x)) + \kappa \|f\|_{\mathcal{F}}^2\right)^{1/2} \left(\text{var}(g(y)) + \kappa \|g\|_{\mathcal{G}}^2\right)^{1/2}}, \quad (6)$$

although this requires an additional parameter κ , which complicates the model selection problem. As the number of observations increases, κ must approach zero to ensure consistency of the estimated KCC, and of the associated functions f and g that achieve the supremum. The rate of decrease of κ for consistency of KCC is derived by Leurgans et al. (1993) (for RKHSs based on spline kernels), and the rate required for consistency in the L_2 norm of f and g is obtained by Fukumizu et al. (2005) (for all RKHSs).

An alternative solution to the problem described in Lemma 9 is given by Kuss (2001), in which the projection directions used to compute the canonical correlations are expressed in terms of a more restricted set of basis functions, rather than the respective subspaces of \mathcal{F} and \mathcal{G} spanned by the entire set of mapped observations. These basis functions can be chosen using kernel PCA, for instance.

Finally, we show that the regularised kernel canonical correlation is a measure of independence, as long as the functions attaining the supremum have bounded variance.

Theorem 10 (KCC ($\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G}, \kappa) = 0$ only at independence for universal kernels) Denote by \mathcal{F} and \mathcal{G} RKHSs with universal kernels on the compact metric spaces \mathcal{X} and \mathcal{Y} , respectively, and assume that $\text{var}(f(x)) < \infty$ and $\text{var}(g(y)) < \infty$. Then $\text{KCC}(\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G}, \kappa) = 0$ if and only if x, y are independent.

Proof The proof is almost identical to the proof of Theorem 6. First, it is clear that x and y being independent implies $\text{KCC}(\mathbf{P}_{x,y}; \mathcal{F}, \mathcal{G}, \kappa) = 0$. Next, assume $\text{COCO}(\mathbf{P}_{x,y}; B(\mathcal{X}), B(\mathcal{Y})) = c$ for $c > 0$. We can then define $f^* \in \mathcal{F}$ and $g^* \in \mathcal{G}$ as before, such that

$$\text{cov}(f^*(x), g^*(y)) \geq \frac{c}{4}.$$

Finally, assuming $\text{var}(f(x))$ and $\text{var}(g(y))$ to be bounded, we get

$$\begin{aligned} & \text{cov} \left(\frac{f^*(x)}{\left(\text{var}(f^*(x)) + \kappa \|f^*\|_{\mathcal{F}}^2\right)^{1/2}}, \frac{g^*(y)}{\left(\text{var}(g^*(y)) + \kappa \|g^*\|_{\mathcal{G}}^2\right)^{1/2}} \right) \\ & \geq \frac{c}{4 \left(\text{var}(f^*(x)) + \kappa \|f^*\|_{\mathcal{F}}^2\right)^{1/2} \left(\text{var}(g^*(y)) + \kappa \|g^*\|_{\mathcal{G}}^2\right)^{1/2}} \\ & > 0. \end{aligned}$$

The requirement of bounded variance is not onerous: indeed, as in the case of the covariance operator, we are guaranteed that $\text{var}(f(x))$ and $\text{var}(g(y))$ are bounded when k and l are bounded. ■

3. Kernel Approximations to the Mutual Information

In this section, we investigate approximations to the mutual information that can be used for measuring independence. Our main results are in Section 3.1. We present the kernel mutual information (KMI) in Definition 14, and prove it to be zero if and only if the empirical COCO is zero (Theorem 15), which justifies using the KMI as a measure of independence. We then show the KMI upper bounds a Parzen window estimate of the mutual information near independence (Theorem 16). An important property of this bound is that it does *not* require numerical integration, or indeed any space partitioning or grid-based approximations (see e.g. Paninski (2003) and references therein). Rather, we are able to obtain a closed form expression when the grid⁸ becomes infinitely fine.

We should emphasise at this point an important distinction between the KMI and KGV on one hand, and COCO and the KCC on the other. We recall that the empirical COCO in Lemma 3 is a finite sample estimate of the population quantity in Definition 2, and the empirical KCC in Lemma 8 has a population equivalent in Definition 7 (convergence of the empirical estimates to the population quantities is guaranteed in both cases, as described in the discussion of Section 2). The KMI and KGV, on the other hand, are bounds on particular sample-based quantities, and are *not* defined here with respect to corresponding population expressions. That said, the KGV appears to be a regularised empirical estimate of the mutual information for Gaussian processes of Baker

⁸. Introduced in the discrete approximation to the mutual information

(1970), although to our knowledge the convergence of the KGV to this population quantity is not yet established.

In Section 3.2, we derive generalisations of COCO and the KMI to more than two univariate random variables. We prove the high dimensional COCO and KMI are zero if and only if the associated pairwise empirical constrained covariances are zero, which makes them suited for application in ICA (see Theorem 24).

3.1 The KMI, the KGV, and the Mutual Information

Three intermediate steps are required to obtain the KMI from the mutual information: an approximation to the MI which is accurate near independence, a Parzen window estimate of this approximation, and finally a bound on the empirical estimate. We begin in Section 3.1.1 by introducing the mutual information between two multivariate Gaussian random variables, for which a closed form solution exists. In Section 3.1.2, we describe a discrete approximation to the mutual information between two continuous, univariate random variables with an arbitrary joint density function, which is defined via a partitioning of the continuous space into a uniform grid of bins; it is well established that this approximation approaches the continuous mutual information as the grid becomes infinitely fine (Cover and Thomas, 1991). We then show in Section 3.1.3 that the discrete mutual information may be approximated by the Gaussian mutual information (GMI), by doing a Taylor expansion of both quantities to second order around independence.

We next address how to go about estimating this Gaussian approximation of the discrete mutual information, given observations drawn according to some probability density. In Section 3.1.4, we derive a Parzen window estimate of the GMI. Next, in Section 3.1.5, we give an upper bound on the empirical GMI, which constitutes the kernel mutual information. Finally, we demonstrate in Section 3.1.6 that the regularised kernel generalised variance (KGV) proposed by Bach and Jordan (2002a) is an upper bound on the KMI, and hence on the Gaussian mutual information, under certain circumstances. A comparison with the link originally proposed between the KGV and the mutual information is given in Appendix B.2.

3.1.1 MUTUAL INFORMATION BETWEEN TWO MULTIVARIATE GAUSSIAN RANDOM VARIABLES

We begin by introducing the Gaussian mutual information and its relation with the canonical correlation. Thus, the present section should be taken as background material which we will refer back to in the discussion that follows. Cover and Thomas (1991) provide a more detailed and general discussion of these principles. If $\mathbf{x}_G, \mathbf{y}_G$ are Gaussian random vectors⁹ in $\mathbb{R}^{l_x}, \mathbb{R}^{l_y}$ respectively, with joint covariance matrix $\mathbf{C} := \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{xy}^\top & \mathbf{C}_{yy} \end{bmatrix}$, then the mutual information between them can be written

$$I(\mathbf{x}_G; \mathbf{y}_G) = -\frac{1}{2} \log \left(\frac{|\mathbf{C}|}{|\mathbf{C}_{xx}| |\mathbf{C}_{yy}|} \right), \tag{7}$$

where $|\cdot|$ is the determinant. We note that the Gaussian mutual information takes the distinctive form of a log ratio of determinants: we will encounter this expression repeatedly in the subsequent

9. The subscripts G are used to emphasise that $\mathbf{x}_G, \mathbf{y}_G$ are Gaussian; this notation is introduced here to make the reasoning clearer in subsequent sections.

reasoning, under various guises. For this reason, we now present a theorem which describes several alternative expressions for this ratio.

Theorem 11 (Ratio of determinants) *Given a partitioned matrix*¹⁰

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \succ \mathbf{0}, \quad (8)$$

we can write

$$\begin{aligned} \frac{\left| \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \right|}{|\mathbf{A}||\mathbf{C}|} &= \left| \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1/2}\mathbf{B}\mathbf{C}^{-1/2} \\ \mathbf{C}^{-1/2}\mathbf{B}^\top\mathbf{A}^{-1/2} & \mathbf{I} \end{bmatrix} \right| \\ &= \left| \mathbf{I} - \mathbf{A}^{-1/2}\mathbf{B}\mathbf{C}^{-1}\mathbf{B}^\top\mathbf{A}^{-1/2} \right| \\ &= \prod_i (1 - \rho_i^2) \\ &> 0 \end{aligned}$$

where ρ_i are the singular values of $\mathbf{A}^{-1/2}\mathbf{B}\mathbf{C}^{-1/2}$ (i.e. the positive square root of the eigenvalues of $\mathbf{A}^{-1/2}\mathbf{B}\mathbf{C}^{-1}\mathbf{B}^\top\mathbf{A}^{-1/2}$). Alternatively, we can write ρ_i as the positive solutions to the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{0} \end{bmatrix} \mathbf{a}_i = \rho_i \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \mathbf{a}_i.$$

The proof is in Appendix A.2. Using this result, we may rewrite (7) as

$$I(\mathbf{x}_G; \mathbf{y}_G) = -\frac{1}{2} \log \left(\prod_i (1 - \rho_i^2) \right), \quad (9)$$

where ρ_i are the singular values of $\mathbf{C}_{xx}^{-1/2}\mathbf{C}_{xy}\mathbf{C}_{yy}^{-1/2}$; or alternatively, the positive solutions to the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{xy}^\top & \mathbf{0} \end{bmatrix} \mathbf{a}_i = \rho_i \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy} \end{bmatrix} \mathbf{a}_i. \quad (10)$$

In this final configuration, it is apparent that ρ_i are the canonical correlates of the Gaussian random variables \mathbf{x}_G and \mathbf{y}_G . We note that the definition of the Gaussian mutual information provided by (9) and (10) holds even when \mathbf{C} does not have full rank (which indicates that $[\mathbf{x}_G^\top \ \mathbf{y}_G^\top]^\top$ spans a subspace of $\mathbb{R}^{l_x+l_y}$), since for $\mathbf{C} \succeq \mathbf{0}$ we require \mathbf{C}_{xy} to have the same nullspace as \mathbf{C}_{yy} , and \mathbf{C}_{xy}^\top to have the same nullspace as \mathbf{C}_{xx} . Alternatively, we could make a change of variables to a lower dimensional space in which the resulting covariance has full rank, and then use the ratio of determinants (7) with this new covariance.

10. We use $\mathbf{X} \succ \mathbf{0}$ to indicate that \mathbf{X} is positive definite.

3.1.2 MUTUAL INFORMATION BETWEEN DISCRETISED UNIVARIATE RANDOM VARIABLES

In this section, and in the sections that follow, we consider only the case where \mathcal{X} and \mathcal{Y} are closed, bounded subsets of \mathbb{R} , and require $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to have the joint density $\mathbf{p}_{x,y}$ (this is by contrast with the discussion in Section 2, in which \mathcal{X} and \mathcal{Y} were defined simply as separable metric spaces, and the measure $\mathbf{P}_{x,y}$ did not necessarily admit a density). We will also assume $\mathcal{X} \times \mathcal{Y}$ represents the support of $\mathbf{p}_{x,y}$. The present section introduces a discrete approximation to the mutual information between x and y , as described by Cover and Thomas (1991). Consider a grid of size $l_x \times l_y$ over $\mathcal{X} \times \mathcal{Y}$. Let the indices i, j denote the point $(q_i, r_j) \in \mathcal{X} \times \mathcal{Y}$ on this grid, and let $\mathbf{q} = (q_1, \dots, q_{l_x}), \mathbf{r} = (r_1, \dots, r_{l_y})$ be the complete sequences of grid coordinates. Assume, further, that the spacing between points along the x and y axes is respectively Δ_x and Δ_y (the bins being evenly spaced). We define two multinomial random variables \hat{x}, \hat{y} with a distribution $\mathbf{P}_{\hat{x}, \hat{y}}(i, j)$ over the grid (the complete $l_x \times l_y$ matrix of such probabilities is \mathbf{P}_{xy}); this corresponds to the probability that x, y is within a small interval surrounding the grid position q_i, r_j , so

$$\begin{aligned} \mathbf{P}_{\hat{x}}(i) &= \int_{q_i}^{q_i+\Delta_x} \mathbf{p}_x(x) dx, & \mathbf{P}_{\hat{y}}(j) &= \int_{r_j}^{r_j+\Delta_y} \mathbf{p}_y(y) dy, \\ \mathbf{P}_{\hat{x}, \hat{y}}(i, j) &= \int_{q_i}^{q_i+\Delta_x} \int_{r_j}^{r_j+\Delta_y} \mathbf{p}_{x,y}(x, y) dx dy. \end{aligned}$$

Thus $\mathbf{P}_{\hat{x}, \hat{y}}(i, j)$ is a discretisation of $\mathbf{p}_{x,y}$. Finally, we denote as \mathbf{p}_x the vector for which $(\mathbf{p}_x)_i = \mathbf{P}_{\hat{x}}(i)$, with a similar \mathbf{p}_y definition. The mutual information between \hat{x} and \hat{y} is defined as

$$I(\hat{x}; \hat{y}) = \sum_{i=1}^{l_x} \sum_{j=1}^{l_y} \mathbf{P}_{\hat{x}, \hat{y}}(i, j) \log \left(\frac{\mathbf{P}_{\hat{x}, \hat{y}}(i, j)}{\mathbf{P}_{\hat{x}}(i) \mathbf{P}_{\hat{y}}(j)} \right). \quad (11)$$

It is well known that $I(x, y)$ is the limit of $I(\hat{x}; \hat{y})$ as the discretisation becomes infinitely fine (Cover and Thomas, 1991, Section 9.5).

3.1.3 MULTIVARIATE GAUSSIAN APPROXIMATION TO THE DISCRETISED MUTUAL INFORMATION

In this section, we draw together results from the two previous sections, showing it is possible to approximate the *discrete* mutual information in Section 3.1.2 with a *Gaussian* mutual information between vectors of sufficiently high dimension, as long as we are close to independence. The results in this section are due to Bach and Jordan (2002a), although the proof of (18) below is novel. We begin by defining an equivalent multidimensional representation $\check{\mathbf{x}}, \check{\mathbf{y}}$ of \hat{x}, \hat{y} in the previous section, where $\check{\mathbf{x}} \in \mathbb{R}^{l_x}$ and $\check{\mathbf{y}} \in \mathbb{R}^{l_y}$, such that $\hat{x} = i$ is equivalent to $(\check{\mathbf{x}})_i = 1$ and $(\check{\mathbf{x}})_{j: j \neq i} = 0$. To be precise, we define the functions¹¹

$$\mathfrak{R}_i(x) = \begin{cases} 1 & x \in [q_i, q_i + \Delta_x) \\ 0 & \text{otherwise} \end{cases}, \quad \mathfrak{R}_j(y) = \begin{cases} 1 & y \in [r_j, r_j + \Delta_y) \\ 0 & \text{otherwise} \end{cases},$$

such that

$$\mathbf{E}_x(\mathfrak{R}_i(x)) = \mathbf{E}_x((\check{\mathbf{x}})_i) = \int_{-\infty}^{\infty} \mathfrak{R}_i(x) \mathbf{p}_x(x) dx = \mathbf{P}_{\hat{x}}(i)$$

11. Note that we do *not* require $\Delta_x = \Delta_y$: thus the functions $\mathfrak{R}_i(x)$ and $\mathfrak{R}_j(y)$ below may not be identical (the argument of the function specifies whether Δ_x or Δ_y is used, to simplify notation).

and

$$\mathbf{E}_{x,y}(\mathfrak{R}_i(x)\mathfrak{R}_j(y)) = \mathbf{E}_{x,y}\left(\left(\check{\mathbf{x}}\right)_i\left(\check{\mathbf{y}}\right)_j\right) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathfrak{R}_i(x)\mathfrak{R}_j(y)\mathbf{p}_{x,y}(x,y)dxdy = \mathbf{P}_{\hat{x},\hat{y}}(i,j).$$

A specific instance of the second formula is when $y = x$, $\mathfrak{R}_i(x) = \mathfrak{R}_i(y)$, and $\mathbf{p}_{x,y}(x,y) = \delta_x(y)\mathbf{p}_x(x)$, where $\delta_x(y)$ is a delta function centred at x . Then

$$\begin{aligned}\mathbf{E}_x(\mathfrak{R}_i(x)\mathfrak{R}_j(x)) &= \mathbf{E}_x\left(\left(\check{\mathbf{x}}\check{\mathbf{x}}^\top\right)_{i,j}\right) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathfrak{R}_i(x)\mathfrak{R}_j(y)\mathbf{p}_x(x)\delta_x(y)dxdy \\ &= \begin{cases} \mathbf{P}_{\hat{x}}(i) & i = j \\ 0 & \text{otherwise} \end{cases}.\end{aligned}$$

In summary,

$$\mathbf{E}_{x,y}\left(\check{\mathbf{x}}\check{\mathbf{y}}^\top\right) = \mathbf{P}_{xy} \quad (12)$$

$$\mathbf{E}_x(\check{\mathbf{x}}) = \mathbf{p}_x \quad (13)$$

$$\mathbf{E}_x\left(\check{\mathbf{x}}\check{\mathbf{x}}^\top\right) = \mathbf{D}_x \quad (14)$$

where $\mathbf{D}_x = \text{diag}(\mathbf{p}_x)$. Using these results, it is possible to define the covariances

$$\mathbf{C}_{xy} = \mathbf{E}_{x,y}(\check{\mathbf{x}}\check{\mathbf{y}}^\top) - \mathbf{E}_x(\check{\mathbf{x}})\mathbf{E}_y(\check{\mathbf{y}})^\top = \mathbf{P}_{xy} - \mathbf{p}_x\mathbf{p}_y^\top, \quad (15)$$

$$\mathbf{C}_{xx} = \mathbf{E}_x(\check{\mathbf{x}}\check{\mathbf{x}}^\top) - \mathbf{E}_x(\check{\mathbf{x}})\mathbf{E}_x(\check{\mathbf{x}})^\top = \mathbf{D}_x - \mathbf{p}_x\mathbf{p}_x^\top, \quad (16)$$

$$\mathbf{C}_{yy} = \mathbf{E}_y(\check{\mathbf{y}}\check{\mathbf{y}}^\top) - \mathbf{E}_y(\check{\mathbf{y}})\mathbf{E}_y(\check{\mathbf{y}})^\top = \mathbf{D}_y - \mathbf{p}_y\mathbf{p}_y^\top. \quad (17)$$

We may therefore define Gaussian random variables $\mathbf{x}_G, \mathbf{y}_G$ with the same covariance structure as $\check{\mathbf{x}}, \check{\mathbf{y}}$, and with mutual information given by (7). We prove in Appendix A.3 that the mutual information for this Gaussian case is

$$I(\mathbf{x}_G; \mathbf{y}_G) = -\frac{1}{2}\log\left(\left|\mathbf{I}_y - \left(\mathbf{P}_{xy} - \mathbf{p}_x\mathbf{p}_y^\top\right)^\top \mathbf{D}_x^{-1} \left(\mathbf{P}_{xy} - \mathbf{p}_x\mathbf{p}_y^\top\right) \mathbf{D}_y^{-1}\right|\right), \quad (18)$$

which can also be expressed in the singular value form (9). The relation between (18) and (11) is given in the following lemma, which is proved by Bach and Jordan (2002a, Appendix. B.1).

Lemma 12 (The discrete MI approximates the Gaussian MI near independence)

Let $\mathbf{P}_{\hat{x},\hat{y}}(i,j) = \mathbf{P}_{\hat{x}}(i)\mathbf{P}_{\hat{y}}(j)(1 + \varepsilon_{i,j})$ for an appropriate choice of $\varepsilon_{i,j}$, where $\varepsilon_{i,j}$ is small near independence. Then the second order Taylor expansion of the discrete mutual information in (11) is

$$I(\hat{x}; \hat{y}) \approx \frac{1}{2}\sum_{i=1}^{l_x}\sum_{j=1}^{l_y}\mathbf{P}_{\hat{x}}(i)\mathbf{P}_{\hat{y}}(j)\varepsilon_{i,j}^2,$$

which is equal to the second order Taylor expansion of the Gaussian mutual information in (18), namely

$$I(\mathbf{x}_G; \mathbf{y}_G) \approx \frac{1}{2}\sum_{i=1}^{l_x}\sum_{j=1}^{l_y}\mathbf{P}_{\hat{x}}(i)\mathbf{P}_{\hat{y}}(j)\varepsilon_{i,j}^2.$$

3.1.4 KERNEL DENSITY ESTIMATES OF THE GAUSSIAN MUTUAL INFORMATION

In this section, we describe a kernel density estimate of the approximate mutual information in (18): this is the point at which our reasoning diverges from the approach of Bach and Jordan (2002a). Before proceeding, we motivate this discussion with a short overview of the Parzen window estimate and its properties, as drawn from Silverman (1986); Duda et al. (2001) (this discussion pertains to the general case of multivariate \mathbf{x} , although our application requires only univariate random variables). Given a sample \mathbf{x} of size m , each point x_l of which is assumed generated i.i.d. according to some unknown distribution with density \mathbf{p}_x , the associated Parzen window estimate of this density is written

$$\hat{\mathbf{p}}_x(x) = \frac{1}{m} \sum_{l=1}^m \kappa(x_l - x).$$

The kernel function¹² $\kappa(x_l - x)$ must be a legitimate probability density function, in that it should be correctly normalised,

$$\int_{\mathcal{X}} \kappa(x) dx = 1, \tag{19}$$

and $\kappa(x) \geq 0$. We may rescale the kernel according to $\frac{1}{V_x} \kappa\left(\frac{x}{\sigma_x}\right)$, where the term V_x is needed to preserve (19). Denoting as $V_{x,m}$ the normalisation for a sample size m , then we are guaranteed that the Parzen window estimate converges to the true probability density as long as

$$\begin{aligned} \lim_{m \rightarrow \infty} V_{x,m} &= 0, \\ \lim_{m \rightarrow \infty} mV_{x,m} &= \infty. \end{aligned}$$

This method requires an initial choice of σ_x for the sample size we start with, which can be obtained by cross validation.

We return now to the problem of empirically estimating the mutual information described in Sections 3.1.2 and 3.1.3. Our estimate is described in the following definition.

Definition 13 (Parzen window estimate of the Gaussian mutual information) *A Parzen window estimate of the Gaussian mutual information in (18) is defined as*

$$\hat{I}(\hat{\mathbf{x}}; \hat{\mathbf{y}}) = -\frac{1}{2} \log \left(\prod_{i=1}^{\min(l_x, l_y)} (1 + \hat{\rho}_i)(1 - \hat{\rho}_i) \right), \tag{20}$$

where $\hat{\rho}_i$ are the singular values of

$$\left(\mathbf{D}_l^{(x)}\right)^{-1/2} \left(\mathbf{K}_l \mathbf{H}(\mathbf{L}_l)^\top\right) \left(\mathbf{D}_l^{(y)}\right)^{-1/2}. \tag{21}$$

Of the four matrices in this definition, $\mathbf{D}_l^{(x)}$ is a diagonal matrix of unnormalised Parzen window estimates of \mathbf{p}_x at the grid points,

$$\mathbf{D}_l^{(x)} = \frac{1}{\Delta_x} \begin{bmatrix} \sum_{l=1}^m \kappa(q_1 - x_l) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sum_{l=1}^m \kappa(q_{l_x} - x_l) \end{bmatrix}, \tag{22}$$

12. The reader should not confuse the present kernel with the RKHS kernels introduced earlier. That said, we shall see later that the two kernels are linked.

$\mathbf{D}_l^{(y)}$ is the equivalent diagonal matrix for \mathbf{p}_y ,¹³ and

$$\mathbf{K}_l := \begin{bmatrix} \kappa(q_1 - x_1) & \dots & \kappa(q_1 - x_m) \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \kappa(q_{l_x} - x_1) & \dots & \kappa(q_{l_x} - x_m) \end{bmatrix}, \quad \mathbf{L}_l := \begin{bmatrix} \kappa(r_1 - y_1) & \dots & \kappa(r_1 - y_m) \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \kappa(r_{l_y} - y_1) & \dots & \kappa(r_{l_y} - y_m) \end{bmatrix}, \quad (23)$$

where we write the above in such a manner as to indicate $l_x \gg m$ and $l_y \gg m$.

Details of how we obtained this definition are given in Appendix A.4. The main disadvantage in using this approximation to the mutual information is that it is exceedingly computationally inefficient, in that it requires a kernel density estimate at each point in a fine grid. In the next section, we show that it is possible to eliminate this grid altogether when we take an upper bound.

3.1.5 THE KMI: AN UPPER BOUND ON THE MUTUAL INFORMATION

We now define the kernel mutual information, and show it is both a valid dependence criterion (Theorem 15), and an upper bound on the Parzen GMI in Lemma 13 (Theorem 16).

Definition 14 (The kernel mutual information) *The kernel mutual information is defined as*

$$\begin{aligned} \text{KMI}(z; \mathcal{F}, \mathcal{G}) &:= -\frac{1}{2} \log \left(\left| \mathbf{I} - \mathbf{v}_z^{-2} \tilde{\mathbf{K}} \tilde{\mathbf{L}} \right| \right) \\ &= -\frac{1}{2} \log \left(\prod_i \left(1 - \frac{\gamma_i^2}{\mathbf{v}_z^2} \right) \right), \end{aligned}$$

where γ_i are the non-zero solutions¹⁴ to

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}} \tilde{\mathbf{L}} \\ \tilde{\mathbf{L}} \tilde{\mathbf{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \gamma_i \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}} \\ \tilde{\mathbf{L}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix}, \quad (24)$$

the centred Gram matrices $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$ are defined using RKHS kernels obtained via convolution of the associated Parzen windows,¹⁵

$$k(x_i, x_j) = \int_{\mathcal{X}} \kappa(x_i - q) \kappa(x_j - q) dq \quad \text{and} \quad l(y_i, y_j) = \int_{\mathcal{Y}} \kappa(y_i - r) \kappa(y_j - r) dr,$$

and

$$\mathbf{v}_z = \min \left\{ \min_{j \in \{1 \dots m\}} \sum_{i=1}^m \kappa(x_i - x_j), \min_{j \in \{1 \dots m\}} \sum_{i=1}^m \kappa(y_i - y_j) \right\}.$$

13. As in our Section 3.1.3 definition of $\mathfrak{K}_i(x)$ and $\mathfrak{K}_j(y)$, we use the notation $\kappa(x)$ and $\kappa(y)$ to denote the Parzen windows for the estimates $\hat{\mathbf{p}}_x(x)$ and $\hat{\mathbf{p}}_y(y)$, respectively, even though these may not be identical kernel functions. The argument again indicates which kernel is used.

14. Compare with (4).

15. Recall that $\kappa(x - q)$ may be different from $\kappa(y - r)$, and that the identity of the Parzen window is specified by its argument.

We note that the above definition bears some similarity to the estimate of Pham (2002). That said, we approximate the mutual information, rather than the entropy; in addition, the KMI is computed in the limit of infinitely small grid size, which removes the need for binning. Thus, we retain our original kernel, rather than using a spline kernel in all cases. This allows us greater freedom to choose a kernel density appropriate to the characteristics of the sources.

The KMI inherits the following important property from the constrained covariance.

Theorem 15 (The KMI is zero if and only if the empirical COCO is zero) *The KMI is zero, $\text{KMI}(\mathbf{z}; \mathcal{F}, \mathcal{G}) = 0$, if and only if the empirical constrained covariance is zero, $\text{COCO}(\mathbf{z}; \mathcal{F}, \mathcal{G}) = 0$.*

Proof This theorem follows from the constrained covariance being the largest eigenvalue γ_i of (24). ■

The relation of the KMI to the mutual information is given by the following theorem, which is the main result of Section 3.

Theorem 16 (The KMI upper bounds the GMI) *Assume that $X \times \mathcal{Y}$ is chosen to be the support of $\mathbf{p}_{x,y}$, that $\mathbf{p}_{x,y}$ is bounded away from zero, and that*

$$\begin{aligned} \min_{x \in X} \sum_{i=1}^m \kappa(x - x_i) &\approx \min_{j \in \{1 \dots m\}} \sum_{i=1}^m \kappa(x_i - x_j) \quad \text{and} \\ \min_{y \in \mathcal{Y}} \sum_{i=1}^m \kappa(y - y_i) &\approx \min_{j \in \{1 \dots m\}} \sum_{i=1}^m \kappa(y_i - y_j) \end{aligned}$$

(the expressions above are alternative, unnormalised estimates of $\min_{x \in X} \mathbf{p}_x(x)$ and $\min_{y \in \mathcal{Y}} \mathbf{p}_y(y)$, respectively; the right hand expressions are used so as to obtain the KMI entirely in terms of the sample \mathbf{z}). Then

$$\text{KMI}(\mathbf{z}; \mathcal{F}, \mathcal{G}) \gtrsim \hat{I}(\hat{x}; \hat{y}). \tag{25}$$

This theorem is proved in Appendix A.5. In particular, the approximate nature of the inequality (25) arises from our use of empirical estimates for lower bounds on $\mathbf{p}_x(x)$ and $\mathbf{p}_y(y)$ (see the proof for details).

3.1.6 THE KGV: AN ALTERNATIVE UPPER BOUND ON THE MUTUAL INFORMATION

Bach and Jordan (2002a) propose two related quantities as independence functionals: the kernel canonical correlation (KCC), as discussed in Section 2.4, and the kernel generalised variance (KGV). In this section, we demonstrate that the latter quantity is an upper bound on the KMI under certain conditions. This approach is different to the proof of Bach and Jordan, who employ a limit as the RKHS kernels become infinitely small, and do not make use of Parzen windows. In any event, there may be some problems with this limiting argument: see Appendix B.2 for further discussion. We begin by recalling the definition of the KGV.

Definition 17 (The kernel generalised variance) *The empirical KGV is defined as*

$$\text{KGV}(\mathbf{z}; \mathcal{F}, \mathcal{G}, \theta) = -\frac{1}{2} \log \left(\prod_i (1 - \rho_i^2) \right), \tag{26}$$

where ρ_i are the solutions to the generalised eigenvalue problem¹⁶

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ \tilde{\mathbf{L}}\tilde{\mathbf{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} \theta\tilde{\mathbf{K}}^2 + \nu_z(1-\theta)\tilde{\mathbf{K}} & \mathbf{0} \\ \mathbf{0} & \theta\tilde{\mathbf{L}}^2 + \nu_z(1-\theta)\tilde{\mathbf{L}} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix}, \quad (27)$$

and $\theta \in [0, 1]$.

Next, we demonstrate the link between the KGV and the KMI.

Theorem 18 (The KGV upper bounds the KMI) For all $\theta \in [0, 1]$,

$$\text{KGV}(\mathbf{z}; \mathcal{F}, \mathcal{G}, \theta) \geq \text{KMI}(\mathbf{z}; \mathcal{F}, \mathcal{G}),$$

with equality only at $\theta = 0$, subject to the conditions

$$\nu_z \mathbf{I} - \tilde{\mathbf{K}} \succ 0 \quad \text{and} \quad \nu_z \mathbf{I} - \tilde{\mathbf{L}} \succ 0. \quad (28)$$

This theorem is proved in Appendix A.6. The requirements (28) should be checked at the point of implementation to guarantee a bound, but we are assured of being able to enforce them: for example, when k is the convolution of (properly normalised) Gaussian kernels κ of size σ , then

$$k(x_i, x_j) = \frac{1}{\sqrt{2\pi(2\sigma^2)}} \exp\left(-\frac{1}{2(2\sigma^2)}(x_j - x_i)^2\right),$$

which is a Gaussian with twice the variance and $1/\sqrt{2}$ the peak amplitude of κ . An upper bound on the spectral norm of $\tilde{\mathbf{K}}$ is $\max_j \sum_{i=1}^m k(x_i, x_j)$, which follows from Horn and Johnson (1985, Corollary 6.1.5).¹⁷ In other words, even by this conservative estimate, we are assured there exists a $\sigma > 0$ small enough for (28) to hold (the requirements (28) are also sufficient to guarantee the existence of the KMI, since they cause the argument of the logarithm in Definition 14 to be positive).

3.2 Multivariate COCO and KMI

We now describe how our dependence functionals may be generalised to more than two random variables. Let us define the continuous univariate random variables x_1, \dots, x_n on $\mathcal{X}_1, \dots, \mathcal{X}_n$, with joint distribution $\mathbf{P}_{x_1, \dots, x_n}$. We also define the associated feature spaces $\mathcal{F}_{\mathcal{X}_1}, \dots, \mathcal{F}_{\mathcal{X}_n}$, each with its corresponding kernel (as in the 2 variable case, the kernels may be different). We begin with a generalisation of the concept of constrained covariance. Our expression takes a similar form to that of Bach and Jordan (2002a, Appendix A.3), although they deal with canonical correlations rather than constrained covariances, which changes the discussion in some respects.

Definition 19 (Empirical multivariate COCO) Let $\mathbf{z} := \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be an i.i.d. sample of size m from the joint distribution $\mathbf{P}_{x_1, \dots, x_n}$. The multivariate COCO is defined as

$$\text{COCO}(\mathbf{z}; F_{\mathcal{X}_1}, \dots, F_{\mathcal{X}_n}) := \max_j (|\lambda_j|),$$

16. See (5). Note that Bach and Jordan (2002a) handle the scaling differently: they replace the right hand matrix in (27) with $\begin{bmatrix} \tilde{\mathbf{K}}^2 + \zeta\tilde{\mathbf{K}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{L}}^2 + \zeta\tilde{\mathbf{L}} \end{bmatrix}$ for a regularisation scale ζ . We shall see that the form in (27) guarantees the KGV to upper bound the KMI (and hence $\hat{T}(\hat{x}, \hat{y})$ in (20)).

17. Bearing in mind Lemma 27, and that \mathbf{H} has singular values in $\{1, 0\}$.

where λ_j are the solutions to the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}}_1 \tilde{\mathbf{K}}_2 & \dots & \tilde{\mathbf{K}}_1 \tilde{\mathbf{K}}_n \\ \tilde{\mathbf{K}}_2 \tilde{\mathbf{K}}_1 & \mathbf{0} & \dots & \tilde{\mathbf{K}}_2 \tilde{\mathbf{K}}_n \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{K}}_n \tilde{\mathbf{K}}_1 & \tilde{\mathbf{K}}_n \tilde{\mathbf{K}}_2 & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_{1,j} \\ \mathbf{c}_{2,j} \\ \vdots \\ \mathbf{c}_{n,j} \end{bmatrix} = \lambda_j \begin{bmatrix} \tilde{\mathbf{K}}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{K}}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{K}}_n \end{bmatrix} \begin{bmatrix} \mathbf{c}_{1,j} \\ \mathbf{c}_{2,j} \\ \vdots \\ \mathbf{c}_{n,j} \end{bmatrix}, \quad (29)$$

$\tilde{\mathbf{K}}_i = \mathbf{H}\mathbf{K}_i\mathbf{H}$, and \mathbf{K}_i is the uncentred Gram matrix of the observations \mathbf{x}_i drawn from \mathbf{P}_{x_i} .

This expression is obtained using reasoning analogous to the bivariate empirical COCO in Section 2. The following result justifies using the multivariate COCO as an independence measure.

Lemma 20 (The multivariate COCO measures pairwise independence) *The multivariate constrained covariance is zero if and only if all the empirical pairwise constrained covariances are zero:*

$$\text{COCO}(\mathbf{z}; F_{x_1}, \dots, F_{x_n}) = 0 \text{ iff } \text{COCO}(\mathbf{x}_i, \mathbf{x}_j; F_{x_i}, F_{x_j}) = 0 \text{ for all } i \neq j.$$

We note that although the multivariate COCO only verifies pairwise independence, this is nonetheless sufficient to recover mutually independent sources in the context of linear ICA: see Theorem 24. It is instructive to compare with the KCC-based dependence functional for more than two variables, which uses the smallest eigenvalue of a matrix of correlations (with diagonal terms equal to one, rather than zero), where this correlation matrix has only positive eigenvalues.

We next introduce a generalisation of the kernel mutual information to more than two variables. By analogy with the 2-variable case in Definition 14, we propose the following definition.

Definition 21 (Multivariate KMI) *The kernel mutual information for more than two random variables is defined as*

$$\text{KMI}(\mathbf{z}; \mathcal{F}_{x_1}, \dots, \mathcal{F}_{x_n}) := -\frac{1}{2} \log \prod_{j=1}^{mn} (1 + \check{\lambda}_j), \quad (30)$$

where $v_z \check{\lambda}_j = \lambda_j$, and

$$v_z := \min_{i \in \{1, \dots, n\}} v_{x_i}, \text{ where} \quad (31)$$

$$v_{x_i} := \min_{j \in \{1 \dots m\}} \sum_{l=1}^m \kappa(x_{i,l} - x_{i,j}).$$

For (30) to be defined, it is necessary that $1 + \check{\lambda}_j > 0$ for all j , which is true near independence. The following lemma describes the sense in which the multivariate KMI measures independence.

Lemma 22 (The multivariate KMI measures pairwise independence) *The multivariate KMI is zero if and only if the empirical constrained covariance is zero for every pair of random variables: in other words,*

$$\text{KMI}(\mathbf{z}; \mathcal{F}_{x_1}, \dots, \mathcal{F}_{x_n}) = 0$$

if and only if

$$\text{COCO}(\mathbf{x}_i, \mathbf{x}_j; F_{x_i}, F_{x_j}) = 0$$

for all $i \neq j$.

The proof is in Appendix A.7. We now briefly outline how the dependence functional in (30) relates to the KL divergence. In the case of a Gaussian random vector \mathbf{x}_G , which can be segmented as $\mathbf{x}_G^\top := [\mathbf{x}_{G,1}^\top \ \dots \ \mathbf{x}_{G,n}^\top]$, the KL divergence between the joint distribution of \mathbf{x}_G and the product of the marginal distributions of the $\mathbf{x}_{G,i}$ can be written in terms of the relevant covariance matrices as

$$D_{\text{KL}} \left(\mathbf{p}_{\mathbf{x}_G} \left\| \prod_{i=1}^n \mathbf{p}_{\mathbf{x}_{G,i}} \right. \right) = -\frac{1}{2} \log \left(\frac{|\mathbf{C}|}{\prod_{i=1}^n |\mathbf{C}_{ii}|} \right),$$

where

$$\begin{aligned} \mathbf{C} &= \mathbf{E}_{\mathbf{x}_G} \left(\mathbf{x}_G \mathbf{x}_G^\top \right) - \mathbf{E}_{\mathbf{x}_G} \left(\mathbf{x}_G \right) \mathbf{E}_{\mathbf{x}_G} \left(\mathbf{x}_G^\top \right), \\ \mathbf{C}_{ii} &= \mathbf{E}_{\mathbf{x}_{G,i}} \left(\mathbf{x}_{G,i} \mathbf{x}_{G,i}^\top \right) - \mathbf{E}_{\mathbf{x}_{G,i}} \left(\mathbf{x}_{G,i} \right) \mathbf{E}_{\mathbf{x}_{G,i}} \left(\mathbf{x}_{G,i}^\top \right). \end{aligned}$$

These results should allow us to generalise the reasoning in Section 3.1, substituting the kernel density estimates

$$\begin{aligned} \hat{\mathbf{P}}_{x_i}(x_i) &= \frac{1}{m} \sum_{l=1}^m \kappa(x_{i,l} - x_i), \\ \hat{\mathbf{P}}_{x_1, \dots, x_n}(x_1, \dots, x_n) &= \frac{1}{m} \sum_{l=1}^m \prod_{i=1}^n \kappa(x_{i,l} - x_i), \end{aligned}$$

and applying the bounding technique of Section 3.1.5 to obtain the quantity in (30); this is a reason for our choosing v_z to scale $\check{\lambda}_j$.¹⁸ The details of this generalisation are beyond the scope of the present work.

4. Implementation and Application to ICA

Any practical validation of the independence measures described above is best conducted with respect to some ground truth, in which genuinely independent random variables are tested using the proposed functionals (COCO, KMI). Thus, one test of performance is independent component analysis (ICA): this entails separating independent random variables that have been linearly mixed, using only their property of independence (specifically, we recover the coefficients that describe the linear mixing).

An ICA algorithm using COCO and the KMI comprises two components: the efficient computation of COCO and the KMI, using low rank approximations of the Gram matrices, and gradient descent on the space of linear mixing matrices. These results are summarised from the more detailed discussion by Bach and Jordan (2002a) (although the low rank decomposition is in our case made easier by the absence of the variance term used in the KCC and KGV).

4.1 Efficient Computation of Kernel Dependence Functionals

We note that COCO requires us to determine the eigenvalue of maximum magnitude for an $mn \times mn$ matrix (see (29)), and the KMI is a determinant of an $mn \times mn$ matrix, as specified in (30). For any

18. On a more pragmatic note, the factor v_z generally causes $|\check{\lambda}_j| < |\lambda_j|$, which results in $\text{KMI}(\mathbf{z}; \mathcal{F}_{x_1}, \dots, \mathcal{F}_{x_n})$ being defined further from independence. This is not the only such scaling factor, however.

reasonable sample size m , the cost of these computations is prohibitive. We now describe how the computational complexity of this problem may be substantially reduced. First, we note that any positive (semi)definite matrix can be written $\mathbf{K}_i = \mathbf{Z}_i \mathbf{Z}_i^\top$, where \mathbf{Z}_i is lower triangular: this is known as the Cholesky decomposition. If the eigenvalues of the Gram matrix \mathbf{K}_i decay sufficiently rapidly, however, we may make the approximation

$$\mathbf{K}_i \approx \mathbf{Z}_i \mathbf{Z}_i^\top \tag{32}$$

to the Gram matrix \mathbf{K}_i , where \mathbf{Z}_i is an $m \times d_i$ matrix; the error due to this approach may be measured via the maximum eigenvalue μ_i of $\mathbf{K}_i - \mathbf{Z}_i \mathbf{Z}_i^\top$. The \mathbf{Z}_i are determined via an *incomplete* Cholesky decomposition, in which the smaller pivots are skipped; symmetric permutation of the rows and columns of \mathbf{K}_i is used in the course of this process to increase the accuracy and numerical stability of the approximation. This method is applied by Fine and Scheinberg (2001) to decrease the storage and computational requirements of interior point methods in SVMs, and by Bach and Jordan (2002a) for faster computation of the KGV and KCC (pseudocode algorithms may be found in both references). Once the incomplete Cholesky decomposition is accomplished, we can compute the approximate *centred* Gram matrices according to $\tilde{\mathbf{K}}_i := \mathbf{H} \mathbf{K}_i \mathbf{H} = (\mathbf{H} \mathbf{Z}_i) (\mathbf{H} \mathbf{Z}_i^\top) = \tilde{\mathbf{Z}}_i \tilde{\mathbf{Z}}_i^\top$.

We now show how this low rank decomposition may be used to more efficiently compute the constrained covariance in (29). Substituting

$$\mathbf{d}_{i,j} = \tilde{\mathbf{Z}}_i^\top \mathbf{c}_{i,j},$$

we get

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{Z}}_1 \tilde{\mathbf{Z}}_1^\top \tilde{\mathbf{Z}}_2 & \dots & \tilde{\mathbf{Z}}_1 \tilde{\mathbf{Z}}_1^\top \tilde{\mathbf{Z}}_n \\ \tilde{\mathbf{Z}}_2 \tilde{\mathbf{Z}}_2^\top \tilde{\mathbf{Z}}_1 & \mathbf{0} & \dots & \tilde{\mathbf{Z}}_2 \tilde{\mathbf{Z}}_2^\top \tilde{\mathbf{Z}}_n \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{Z}}_n \tilde{\mathbf{Z}}_n^\top \tilde{\mathbf{Z}}_1 & \tilde{\mathbf{Z}}_n \tilde{\mathbf{Z}}_n^\top \tilde{\mathbf{Z}}_2 & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d}_{1,j} \\ \mathbf{d}_{2,j} \\ \vdots \\ \mathbf{d}_{n,j} \end{bmatrix} = \lambda_j \begin{bmatrix} \tilde{\mathbf{Z}}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{Z}}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{Z}}_n \end{bmatrix} \begin{bmatrix} \mathbf{d}_{1,j} \\ \mathbf{d}_{2,j} \\ \vdots \\ \mathbf{d}_{n,j} \end{bmatrix}.$$

We may premultiply both sides by¹⁹ $\text{diag} \left(\begin{bmatrix} \tilde{\mathbf{Z}}_1^\top & \dots & \tilde{\mathbf{Z}}_n^\top \end{bmatrix} \right)$ without increasing the nullspace of this generalised eigenvalue problem, and we then eliminate $\text{diag} \left(\begin{bmatrix} \tilde{\mathbf{Z}}_1^\top \tilde{\mathbf{Z}}_1 & \dots & \tilde{\mathbf{Z}}_n^\top \tilde{\mathbf{Z}}_n \end{bmatrix} \right)$ from both sides. Making these changes, we are left with

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{Z}}_1^\top \tilde{\mathbf{Z}}_2 & \dots & \tilde{\mathbf{Z}}_1^\top \tilde{\mathbf{Z}}_n \\ \tilde{\mathbf{Z}}_2^\top \tilde{\mathbf{Z}}_1 & \mathbf{0} & \dots & \tilde{\mathbf{Z}}_2^\top \tilde{\mathbf{Z}}_n \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{Z}}_n^\top \tilde{\mathbf{Z}}_1 & \tilde{\mathbf{Z}}_n^\top \tilde{\mathbf{Z}}_2 & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d}_{1,j} \\ \mathbf{d}_{2,j} \\ \vdots \\ \mathbf{d}_{n,j} \end{bmatrix} = \lambda_j \begin{bmatrix} \mathbf{d}_{1,j} \\ \mathbf{d}_{2,j} \\ \vdots \\ \mathbf{d}_{n,j} \end{bmatrix}, \tag{33}$$

which is a much more tractable eigenvalue problem, having dimension $\sum_{i=1}^n d_i$. The same procedure may easily be used to recast (30) as the determinant of an $(\sum_{i=1}^n d_i) \times (\sum_{i=1}^n d_i)$ matrix. We now briefly consider how to choose the rank d_i for a given precision μ_i : this depends on both the density

19. The notation $\text{diag} \left(\begin{bmatrix} \tilde{\mathbf{Z}}_1^\top & \dots & \tilde{\mathbf{Z}}_n^\top \end{bmatrix} \right)$ defines a matrix with blocks $\tilde{\mathbf{Z}}_i^\top$ along the diagonal, and zeros elsewhere. The matrix need not be square, however, and the diagonal is in this case defined in a manner consistent with the asymmetry of the $\tilde{\mathbf{Z}}_i^\top$.

\mathbf{p}_{x_i} and the kernel $k(x_i, x)$. For Gaussian kernels and densities with exponential decay rates, Bach and Jordan (2002a) show the required precision relates to the rank according to $d_i = O(\log(m/\mu_i))$, which demonstrates the slow increase in rank with sample size. In the case of the KGV and KCC, however, the form of the empirical estimate causes eigenvalues less than approximately $10^{-3}m\kappa/2$ to be discarded, which thus serves as a target precision to ensure the \mathbf{Z}_i retain constant rank regardless of m . We also adopt this threshold in our simulations with the Gaussian kernel, although our motivation is purely a reduction of computational cost.

4.2 Independent Component Analysis

We describe the goal of instantaneous independent component analysis (ICA), drawing on the numerous existing surveys of ICA and related methods, including those by Hyvärinen et al. (2001); Lee et al. (2000); Cichocki and Amari (2002); Haykin (1998); as well as the review by Comon (1994) of older literature on the topic. We are given m samples $\mathbf{t} := (\mathbf{t}_1, \dots, \mathbf{t}_m)$ of the n dimensional random vector \mathbf{t} , which are drawn independently and identically from the distribution $\mathbf{P}_{\mathbf{t}}$. The vector \mathbf{t} is related to the random vector \mathbf{s} (also of dimension n) by the linear mixing process

$$\mathbf{t} = \mathbf{B}\mathbf{s}, \quad (34)$$

where \mathbf{B} is a matrix with full rank. We refer to our ICA problem as being *instantaneous* as a way of describing the dual assumptions that any observation \mathbf{t} depends only on the sample \mathbf{s} at that instant, and that the samples \mathbf{s} are drawn independently and identically.

The components s_i of \mathbf{s} are assumed to be mutually independent: this model codifies the assumption that the sources are generated by unrelated phenomena (for instance, one component might be an EEG signal from the brain, while another could be due to electrical noise from nearby equipment). Mutual independence (in the case where the random variables admit probability densities) has the following definition (Papoulis, 1991):

Definition 23 (Mutual independence) *Suppose we have a random vector \mathbf{s} of dimension n . We say that the components s_i are mutually independent if and only if*

$$\mathbf{p}_{\mathbf{s}}(\mathbf{s}) = \prod_{i=1}^n \mathbf{p}_{s_i}(s_i). \quad (35)$$

It follows easily that the random variables are pairwise independent if they are mutually independent; i.e. $\mathbf{p}_{s_i}(s_i)\mathbf{p}_{s_j}(s_j) = \mathbf{p}_{s_i, s_j}(s_i, s_j)$ for all $i \neq j$. The reverse does not hold, however: pairwise independence does not guarantee mutual independence.

Our goal is to recover \mathbf{s} via an estimate \mathbf{W} of the inverse of the matrix \mathbf{B} , such that the recovered vector $\mathbf{x} = \mathbf{W}\mathbf{B}\mathbf{s}$ has mutually independent components.²⁰ For the purpose of simplifying our discussion, we will assume that \mathbf{B} (and hence \mathbf{W}) is an *orthogonal matrix*; in the case of arbitrary \mathbf{B} , the observations must first be decorrelated before an orthogonal \mathbf{W} is applied (Hyvärinen et al., 2001). In our experiments, however, we will deal with general mixing matrices.

20. It turns out that the problem described above is indeterminate in certain respects. For instance, our measure of independence does not change when the ordering of elements in \mathbf{x} is swapped, or when components of \mathbf{x} are scaled by different constant amounts. Thus, source recovery takes place up to these invariances.

Mutual independence is generally difficult to determine. In the case of linear mixing, however, we are able to find a unique optimal unmixing matrix \mathbf{W} using only the *pairwise* independence between elements of \mathbf{x} , which is equivalent to recovering the *mutually* independent terms of \mathbf{s} (up to permutation and scaling). This is due to the following theorem (Comon, 1994, Theorem 11).

Theorem 24 (Mutual independence in linear ICA) *Let \mathbf{s} be a vector of dimension n with mutually independent components, of which at most one is Gaussian, and for which the underlying densities do not contain delta functions. Let \mathbf{x} be a random vector related to \mathbf{s} according to $\mathbf{x} = \mathbf{A}\mathbf{s}$, where \mathbf{A} is an orthogonal $n \times n$ matrix.²¹ Then the properties*

- *The components of \mathbf{x} are pairwise independent*
- *The components of \mathbf{x} are mutually independent*
- *$\mathbf{A} = \mathbf{P}\mathbf{S}$, where \mathbf{P} is a permutation matrix, and \mathbf{S} a diagonal matrix*

are equivalent.

We acknowledge that the application of a general dependence function to linear ICA is not guaranteed to be an optimal non-parametric approach to the problem of estimating the entries in \mathbf{B} —for instance, Samarov and Tsybakov (2004) provide a method that guarantees \sqrt{n} -consistent estimates of the columns of \mathbf{B} under certain smoothness assumptions on the source densities, which is a more natural goal in view of the mixing model (34). Indeed, most specialised ICA algorithms exploit the linear mixing structure of the problem to avoid having to employ a general measure of independence, which makes the task of recovering \mathbf{B} easier. That said, ICA is in general a good benchmark for dependence measures, in that it applies to a problem with a known “ground truth”, and tests that the dependence measures approach zero gracefully as dependent random variables are made to approach independence (through optimisation of the unmixing matrix). In addition, the kernel methods yield better experimental performance than other specialised ICA approaches (including recent state-of-the-art algorithms) in our tests of outlier resistance and musical source separation (see Section 5).

We also note at this point that if elements $\mathbf{t}_i, \mathbf{t}_j$ in the sample \mathbf{t} are *not* drawn independently for $i \neq j$ (for instance, if they are generated by a random process with non-zero correlation between the outputs at different times), then an entirely different set of approaches can be brought to bear (see for instance Belouchrani et al., 1997; Pham and Garat, 1997).²² Although the present study concentrates entirely on the i.i.d. case, we will briefly address random processes with time dependencies in Section 6, when describing possible extensions to our work. Finally, we draw attention to an alternative ICA setting, as described by Cardoso (1998b); Theis (2005), in which \mathbf{s} is partitioned into mutually independent vectors (which might each have internal dependence structure): we wish to recover these vectors following linear mixing. As pointed out by Bach and Jordan (2002a), kernel dependence functionals are well suited to this problem, since they also apply straightforwardly to multivariate random variables: it suffices to define appropriate Gram matrices.

21. For the purposes of ICA, \mathbf{A} combines both the mixing and unmixing processes, *i.e.*, $\mathbf{A} = \mathbf{W}\mathbf{B}$.

22. In particular, it becomes possible to separate Gaussian processes when they are correlated over time.

4.3 Gradient Descent on the Stiefel Manifold

We now describe the method used to minimise our kernel dependence functionals over possible choices of the orthogonal demixing matrix \mathbf{W} . The manifold described by $n \times p$ matrices \mathbf{A} for which $\mathbf{A}^\top \mathbf{A} = \mathbf{I}$, where $n \geq p$, is known as the *Stiefel manifold*. Gradient descent for functions defined on this manifold is described by Edelman et al. (1998), and Bach and Jordan (2002a) applied this technique to kernel ICA. A clear and intuitive explanation of this procedure is also given by Hyvärinen and Plumbley (2002). Let $f(\mathbf{W}, \mathbf{t})$ be the particular dependence functional (COCO or KMI) on which we wish to do gradient descent, where $\mathbf{t} := (\mathbf{t}_1, \dots, \mathbf{t}_m)$ are the whitened, mixed observations. A naive gradient descent algorithm would involve computing the derivative

$$\mathbf{G} := \frac{\partial f(\mathbf{W}, \mathbf{t})}{\partial \mathbf{W}},$$

updating \mathbf{W} according to $\mathbf{W} \rightarrow \mathbf{W} + \mu \mathbf{G}$ (where μ is chosen to minimise $f(\mathbf{W} + \mu \mathbf{G}, \mathbf{t})$), and projecting the resulting matrix back onto the Stiefel manifold. This might not be particularly efficient, however, in that the update can largely be cancelled by the subsequent projection operation. Instead, we attempt to find the direction of steepest descent on the Stiefel manifold, and to perform our update with the constraint that we remain on this manifold. To achieve this, we first describe the set of perturbations to \mathbf{W} that retain the orthogonality of \mathbf{W} , then choose the direction of steepest descent/ascent within this set, and finally give the expression that parameterises the shifts along the geodesic²³ in this direction.

Let $\mathbf{\Delta}$ be a perturbation with small norm to the orthogonal matrix \mathbf{W} , such that $\mathbf{W} + \mathbf{\Delta}$ remains on the Stiefel manifold. For this constraint to hold, we require

$$(\mathbf{W} + \mathbf{\Delta})^\top (\mathbf{W} + \mathbf{\Delta}) = \mathbf{I}, \text{ which implies} \quad (36)$$

$$\mathbf{W}^\top \mathbf{\Delta} + \mathbf{\Delta}^\top \mathbf{W} \approx \mathbf{0}; \quad (37)$$

in other words, $\mathbf{W}^\top \mathbf{\Delta}$ is skew-symmetric. To find the particular $\mathbf{\Delta}$ that gives the direction of steepest change of $f(\mathbf{W}, \mathbf{t})$, we solve

$$\mathbf{\Delta}_{\max} := \arg \max_{\mathbf{\Delta}} f(\mathbf{W} + \mathbf{\Delta}, \mathbf{t}),$$

subject to $\text{tr}(\mathbf{\Delta}^\top \mathbf{\Delta}) = \text{const}$ and (37). This yields

$$\mathbf{\Delta}_{\max} = \mathbf{G} - \mathbf{W} \mathbf{G}^\top \mathbf{W},$$

where the proof is provided by Edelman et al. (1998); Hyvärinen and Plumbley (2002). Finally, if we use q to parameterise displacement along a geodesic in the direction $\mathbf{\Delta}_{\max}$ from an initial matrix $\mathbf{W}(0)$, then the resulting $\mathbf{W}(q)$ is given by

$$\mathbf{W}(q) = \mathbf{W}(0) \exp\left(q \mathbf{W}(0)^\top \mathbf{\Delta}_{\max}\right).$$

As in the implementation of Bach and Jordan (2002a), we determine an approximation of the gradient of $f(\mathbf{W}, \mathbf{t})$ by making small perturbations to \mathbf{W} about each possible Jacobi rotation, and recomputing f for each such perturbation. Gradient descent is then accomplished using a Golden search along this direction of steepest descent.

23. A geodesic represents the shortest path on a manifold between two points; equivalently, the acceleration involved in moving between two points along a geodesic is perpendicular to the manifold when constant velocity is maintained.

Finally, we note that procedures are given by Edelman et al. (1998) to compute the Hessian on the Stiefel manifold, as are implementations of Newton’s method and conjugate gradient descent. In addition, an adaptive algorithm for gradient descent on the Stiefel manifold is proposed by Zhu and Zhang (2002). The application of these methods to improve the performance of our algorithm is beyond the scope of the present work.

4.4 Computational Cost

We conclude this section with a summary of the overall computational cost of ICA based on COCO and the KMI: this analysis draws directly from the assessment of Bach and Jordan (2002a, Section 6), since COCO and the KMI cost effectively the same as the KCC and KGV, respectively. The first step in ICA, which is not discussed here, is the decorrelation of the sources (as described for instance by Hyvärinen et al., 2001), which has a cost $O(mn^2)$. We next consider the cost of computing the multivariate COCO and KMI. In both approaches, each of the n sources requires an estimate of its $m \times m$ Gram matrix using incomplete Cholesky decomposition, which costs $O(md^2)$, where d is the largest rank retained in the computation of the \mathbf{Z}_i in (32): the net cost is $O(mnd^2)$. These \mathbf{Z}_i are then centred and assembled into the matrix in (33), which entails $n(n-1)/2$ operations each costing $O(md^2)$, for an overall cost $O(mn^2d^2)$. COCO is given by the largest eigenvalue of this matrix, and costs $O(n^2d^2)$; the KMI is a determinant, and costs $O(n^3d^3)$.

We compute the gradient of the kernel dependence measures using the method of finite differences (as described in the previous section), which necessitates $n(n-1)/2$ evaluations of the measure used. In each evaluation, we need only compute two incomplete Cholesky decompositions (we cache the remainder); the assembly of the matrix in (33) then entails $2n-3$ matrix products, for an overall cost (Cholesky + matrix assembly for all the Jacobi rotations) of $O(mn^3d^2)$. The eigenvalue computations used to obtain the gradient of COCO cost $O(n^4d^2)$, and the determinants used in the KMI gradient cost $O(n^5d^3)$.

5. Experimental Results on ICA

In this section, we examine the performance of our independence functionals (COCO, KMI) as it compares to the KGV and KCC, when used to address the problem of linear instantaneous ICA. Since the objective is to find an estimate \mathbf{W} of the *inverse* of the mixing matrix \mathbf{B} (the reader is referred to Section 4.2 for a description of the ICA problem), we require a measure of distance between our approximation and the true inverse: this is given by the *Amari divergence*, which is introduced in Section 5.1. Next, in Section 5.2, we present results obtained when separating a range of artificial signals mixed using randomly generated matrices, including cases in which the observations are corrupted by noise. Finally, we describe our attempts at separating artificial mixtures of audio signals representing a number of musical genres. Results are compared with those obtained using standard methods (FastICA, Jade, Infomax) and recent state-of-the-art methods (RADICAL, CFICA), as well as the KCC and KGV.

5.1 Measurement of Performance

We use the Amari divergence, defined by Amari et al. (1996), as an index of ICA algorithm performance: this is an adaptation and simplification of a criterion proposed earlier by Comon (1994).

Note that the properties of this quantity in Lemma 26 were not described by Amari et al. (1996), but follow from the proof of Comon (1994).

Definition 25 (Amari divergence) *Let \mathbf{B} and \mathbf{W} be two $n \times n$ matrices, where \mathbf{B} is the mixing matrix and \mathbf{W} the estimated unmixing matrix (these need not be orthogonal here), and let $\mathbf{D} = \mathbf{WB}$. Then the Amari divergence between \mathbf{B} and \mathbf{W} is*

$$\mathcal{D}(\mathbf{WB}) = \frac{100}{2n(n-1)} \sum_{i=1}^n \left(\frac{\sum_{j=1}^n |d_{i,j}|}{\max_j |d_{i,j}|} - 1 \right) + \frac{1}{2n(n-1)} \sum_{j=1}^n \left(\frac{\sum_{i=1}^n |d_{i,j}|}{\max_i |d_{i,j}|} - 1 \right).$$

Although this measure is not, strictly speaking, a distance metric for general matrices \mathbf{B}, \mathbf{W} , it nonetheless possesses certain useful properties, as shown below.

Lemma 26 (Properties of the Amari divergence) *The Amari divergence $\mathcal{D}(\mathbf{WB})$ between the $n \times n$ matrices \mathbf{B}, \mathbf{W} has the following properties:*

- $0 \leq \mathcal{D}(\mathbf{WB}) \leq 100$. *The factor of 100 is not part of the original definition of Amari et al. (1996), who defined the Amari divergence on $[0, 1]$. In our experiments, however, the Amari divergence was generally small, and we scaled it by 100 to make the results tables more readable.*
- *Let \mathbf{P} be an arbitrary permutation matrix (a matrix with a single 1 in each row and column, and with remaining entries 0), and \mathbf{S} be a diagonal matrix of non-zero scaling factors. Then $\mathbf{W} = \mathbf{B}^{-1}$ if and only if $\mathcal{D}(\mathbf{WB}) = 0$, or equivalently $\mathcal{D}(\mathbf{WBSP}) = 0$ or $\mathcal{D}(\mathbf{SPWB}) = 0$.*

The final property in the above Lemma is particularly useful in the context of ICA, since it causes our performance measure to be invariant to output ordering ambiguity once the sources have been demixed (see Theorem 24).

5.2 Experiments and Performance Assessment

Since our main purpose is to compare the performance with that reported by Bach and Jordan (2002a), we generated our test distributions independently following their descriptions. A list of the distributions used in our experiments, and their respective kurtoses, is given in Table 3. While these distributions represent a broad range of behaviours, we note that negative kurtoses predominate, which should be borne in mind when evaluating performance. We used the KGV and KCC Matlab implementations downloadable from (Bach and Jordan) (thus, we employ the KGV as originally defined by Bach and Jordan (2002a), and not the version described in Section 3.1.6). The precision of the incomplete Cholesky decomposition, used to approximate the Gram matrices for the kernel dependence functionals, was set at $\eta := \epsilon n$; our choice of ϵ represents a tradeoff between accuracy and computation speed. Unless otherwise specified, the kernel algorithm results were refined in a “polishing step”, in which the kernel size was halved upon convergence, and the gradient descent procedure recommenced with this smaller kernel. This polishing was carried out since the larger kernel size results in the kernel dependence measures being a smoother function of the estimated unmixing matrix, making it easier to find the global minimum; but making the location of this global minimum less precise than obtained with a smaller kernel. The polishing step usually caused a measurable improvement in our results.

As well as the kernel algorithms, we compare with three standard ICA methods: FastICA (Hyvärinen et al., 2001), Jade (Cardoso, 1998a), and Infomax (Bell and Sejnowski, 1995); and two more sophisticated methods, neither of them based on kernels: RADICAL (Learned-Miller and Fisher III, 2003), which uses order statistics to obtain entropy estimates; and characteristic function based ICA (CFICA) (Chen and Bickel, 2004).²⁴ It was recommended to run the CFICA algorithm with a good initialising guess; we used RADICAL for this purpose. All kernel algorithms were initialised using Jade (except for the 16 source case, where FastICA was used due to its more stable output). RADICAL is based on an exhaustive grid search over all the Jacobi rotations, and does not require an initial guess. In the case of FastICA, we used the nonlinearity most appropriate to the signal characteristics: this was generally the kurtosis based contrast, since the predominantly negative kurtoses in Table 3 made this a good choice (see Hyvärinen et al., 2001). In some experiments, however, the kurtosis was unsuited to the source characteristics, in which case we signal our alternative choice of nonlinearity. The Infomax algorithm selects its contrast automatically based on the super- or sub-Gaussianity of the signal, and does not require manual contrast choice. Likewise Jade uses only a kurtosis-based contrast, and thus does not require the user to choose a demixing function.

We begin with a brief investigation into the form taken by the various kernel dependence functionals for a selection of the data in Table 3. Contours of the KGV, COCO, KMI, and Amari divergence are plotted in Figure 1, which describes the demixing of samples from three distributions, combined using a product of known Jacobi rotations. All kernel functionals in this demonstration were computed with a Gaussian RBF kernel,

$$k_G(x, x') = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \|x - x'\|^2\right). \quad (38)$$

We observe that each of the functionals exhibits local minima at locations distant from independence, but that each possesses a “basin of attraction” in the vicinity of the correct answer. Moreover, we note that each of the functionals is smooth (given the choice of kernel size), and that the global minima are fairly symmetric. For these reasons, the gradient descent algorithm described in Section 4.3 should converge rapidly to the global optimum, given a reasonable initialisation point. Our solution method differs from that of Bach and Jordan (2002a), however, in that we generally use Jade (unless specified otherwise) to initialise the kernel functionals (COCO, KCC, KGV, KMI), whereas Bach and Jordan only do this when separating large numbers of signals (in most cases, they initialise using a one-unit kernel dependence functional with deflation, and with a less costly polynomial kernel). For more than two signals, this process is repeated several times, starting from different initialising matrices. While Jade is less computationally costly as an initialisation method, it might be less reliable in certain cases (where the sources are near-Gaussian, or when a large number of outliers exist due to noise, both of which can cause Jade to misconverge).

5.3 General Mixtures of Artificial Data

We now describe the ICA experiments performed with the distributions in Table 3, where the Amari divergence is used to measure the closeness of the estimated mixing matrix to the true matrix.

24. We are aware that Chen and Bickel propose an alternative algorithm, “efficient ICA”. We did not include results from this algorithm in our experiments, since it is unsuited to mixtures of Gaussians (which have fast decaying tails) and discontinuous densities (such as the uniform density on a finite interval), which both occur in our benchmark set.

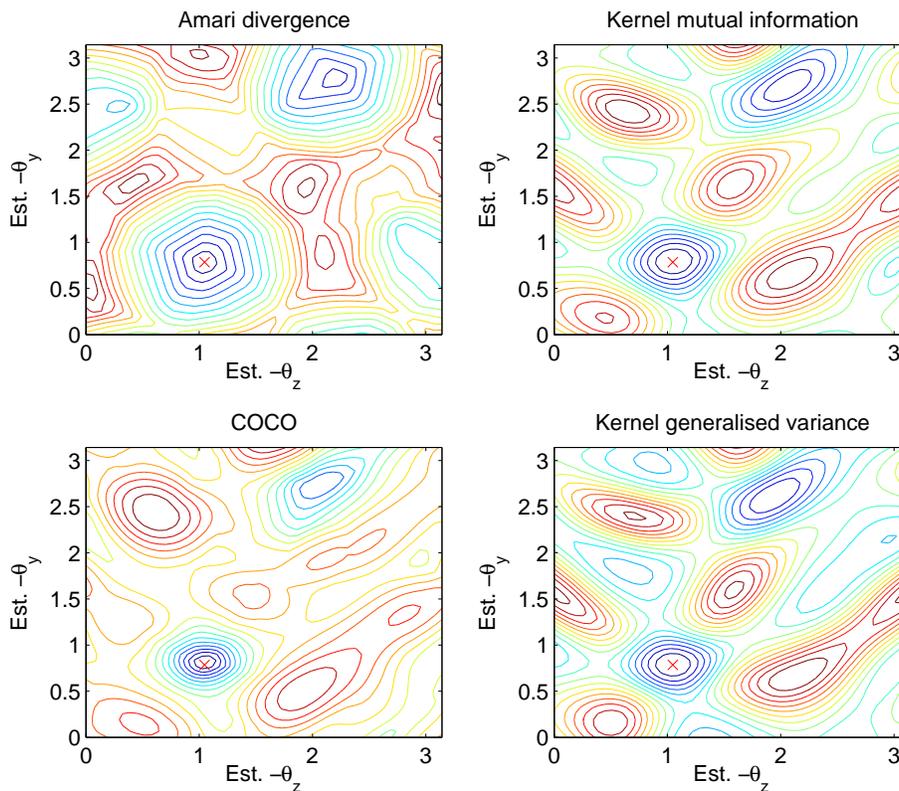


Figure 1: Contour plots of kernel independence functionals. Top left: Amari divergence. Top right: kernel mutual information. Bottom left: constrained covariance. Bottom right: kernel generalised variance. Three signals of length 1000 and with respective distributions g , k , and q (this choice was random) were combined using a 3×3 orthogonal rotation matrix. This matrix was expressed as a product of Jacobi rotations $\mathbf{B} = \mathbf{R}_z(\theta_z)\mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x)$, where $\theta_x = -\pi/6$, $\theta_y = -\pi/4$, and $\theta_z = -\pi/3$; the subscript denotes the axis about which the rotation occurs. An estimate $\mathbf{W} = \mathbf{R}_x(-\theta_x)\mathbf{R}_y(\hat{\theta}_y)\mathbf{R}_z(\hat{\theta}_z)$ of \mathbf{B}^{-1} was made, in which $\hat{\theta}_y$ and $\hat{\theta}_z$ took values in the range $[0, \pi]$. The red “x” in each plot is located at the coordinates $(-\hat{\theta}_z, -\hat{\theta}_y)$ corresponding to the optimal estimate of \mathbf{B} . A Gaussian kernel of size $\sigma^2 = 1$ was used in all cases, and $\kappa = 10^{-3}$ for the KGV.

Label	Definition	Kurtosis
a	Student's t distribution, 3 DOF	∞
b	Double exponential	3.00
c	Uniform	-1.20
d	Students's t distribution, 5 DOF	6.00
e	Exponential	6.00
f	Mixture, 2 double exponentials	-1.70
g	Symmetric mixture 2 Gauss., multimodal	-1.85
h	Symmetric mixture 2 Gauss., transitional	-0.75
i	Symmetric mixture 2 Gauss., unimodal	-0.50
j	Asymm. mixture 2 Gauss., multimodal	-0.57
k	Asymm. mixture 2 Gauss., transitional	-0.29
l	Asymm. mixture 2 Gauss., unimodal	-0.20
m	Symmetric mixture 4 Gauss., multimodal	-0.91
n	Symmetric mixture 4 Gauss., transitional	-0.34
o	Symmetric mixture 4 Gauss., unimodal	-0.40
p	Asymm. mixture 4 Gauss., multimodal	-0.67
q	Asymm. mixture 4 Gauss., transitional	-0.59
r	Asymm. mixture 4 Gauss., unimodal	-0.82

Table 3: Labels of distributions used, and their respective kurtoses. All distributions have zero mean and unit variance.

Kernels used include the Gaussian RBF kernel in (38), and the Laplace kernel,

$$k_L(x, x') = \frac{\lambda}{2} \exp(-\lambda \|x - x'\|).$$

We combined the independent sources using random mixing matrices, with condition numbers between 1 and 2, and then whitened the resulting observations before estimating the orthogonal de-mixing matrix.²⁵

Our first experiment consisted in de-mixing data drawn independently from 2-16 sources chosen at random with replacement from Table 3. Results are given in Table 4. The KMI with Gaussian kernel matches or exceeds KGV performance in the final four experiments; and, with the Laplace kernel, in five of the seven experiments. Moreover, the KMI yields performance statistically indistinguishable from RADICAL in four of the seven experiments.²⁶ On the other hand, the KGV outperforms the KMI in the first and third case, where the number m of samples is small (although in the $n = 4, m = 1000$ case, the difference is not statistically significant). The superior performance of the Laplace kernel compared with the Gaussian may be due to its slower decaying spectrum, which allows dependence encoded at higher frequencies in the source density to induce a greater departure of COCO from zero (making this dependence easier to detect): see Gretton et al. (2005b, Section 4.2). The Laplace kernel has a greater computational cost, however, since the eigenvalues of the associated Gram matrices decay more slowly than for the Gaussian kernel, necessitating the use of a higher rank in the incomplete Cholesky decomposition to maintain good performance. Finally, the extended Infomax algorithm seems unable to separate the signals in 250 sample, 2 signal case: the Amari divergence was spread almost uniformly over the range $[0, 100]$.

5.4 Performance on Difficult Artificial Problems

In our next experiment, we investigated the effect of outlier noise added to the observations. We selected two generating distributions from Table 3, randomly and with replacement. After combining these signals with a randomly generated matrix with condition number between 1 and 2, we generated a varying number of outliers by adding ± 5 (with equal probability) to *both* signals at random locations. All kernels used were Gaussian with size $\sigma = 1$; Laplace kernels resulted in decreased performance for this noisy data. In the case of COCO, this can be explained by functions in the Laplace RKHS having less penalisation at high frequencies, causing the functions attaining the supremum in Definition 2 to adapt to (and be affected by) outliers to a greater degree than functions in the Gaussian RKHS (the KMI is also subject to this effect). Results are shown in the left hand plot in Figure 2. Note that we used $\kappa = 0.11$ for the KGV and KCC in this plot, which is an order of magnitude above the level recommended by Bach and Jordan (2002a): this resulted in an improvement in performance (broadly speaking, an increase in κ causes the KGV to approach the KMI, and

25. We did not use simple orthogonal matrices to mix our sources, since this would have lowered the variance in our estimate of \mathbf{W} , making the problem (slightly) easier than that of estimating a truly random mixing matrix (Cardoso, 1998a).

26. The mean performance of the various methods, both kernel and otherwise, is affected in some experiments by a small number of misconverged results with large Amari divergence (although misconvergence of the kernel methods does not always correspond to misconvergence of the Jade initialisation). These results may arise from diversion to local minima, causing an increase in the overall mean Amari divergence that does not reflect the typical behaviour of the kernel algorithms. Such outliers occur less often, or not at all, at larger sample sizes, as can be seen by the decreased variance in these cases.

the KCC to approach COCO).²⁷ It is clear that the kernel methods substantially outperform both the standard and recent alternatives in outlier resistance (we omitted the remaining standard methods, since their performance was worse than FastICA).

An additional experiment was also carried out on the same data, to test the sensitivity of the KCC and KGV to the choice of the regularisation constant κ . We observe in the right hand plot of Figure 2 that too small a κ can cause severe underperformance for the KCC and KGV. On the other hand, κ is required to be small for good performance at large sample sizes in Table 4. A major advantage of COCO and the KMI is that these do not require any additional tuning beyond the selection of a kernel.

Our third experiment addresses the effects of low kurtosis, since many ICA methods rely (sometimes implicitly, through their choice of nonlinearity) on the kurtosis as an index of signal independence. Two samples were drawn from a single distribution, consisting of a mixture of two Gaussians with means $+5$ and -5 and unit variance, with a selection of mixture weights chosen such that, following normalisation of the overall sample to zero mean and unit variance, the (empirical) kurtosis took on a range of positive, near-zero, and negative values. Results are given in Figure 3. All kernel based methods were unaffected by near-zero kurtosis, as were CFICA and RADICAL; the remaining ICA methods do less well (Infomax was omitted since it performed worse than Jade).

27. The results presented here for the KCC and KGV also improve on those of Learned-Miller and Fisher III (2003); Bach and Jordan (2002a) since they include a polishing step for the KCC and KGV, which was not carried out in these earlier studies.

n	m	Rep.	Fica	Jade	Imax	CFICA	RAD	KCC	COCO(g)	COCO(l)	KGV	KMI(g)	KMI(l)
2	250	1000	10.5±0.4	9.5±0.4	44.4±1	7.2±0.3	5.4±0.2	7.0±0.3	7.8±0.3	7.0±0.3	5.3±0.2	6.0±0.2	5.7±0.2
2	1000	1000	6.0±0.3	5.1±0.2	11.3±0.6	3.2±0.1	2.4±0.1	3.3±0.1	3.5±0.1	2.9±0.1	2.3±0.1	2.6±0.1	2.3±0.1
4	1000	100	5.7±0.4	5.6±0.4	13.3±1	3.3±0.2	2.5±0.1	4.5±0.4	4.2±0.3	4.6±0.6	3.1±0.6	4.0±0.7	3.5±0.7
4	4000	100	3.1±0.2	2.3±0.1	5.9±0.7	1.5±0.1	1.3±0.1	2.4±0.5	1.9±0.1	1.6±0.1	1.4±0.1	1.4±0.05	1.2±0.05
8	2000	50	4.1±0.2	3.6±0.2	9.3±0.9	2.4±0.1	1.8±0.1	4.8±0.9	3.7±0.9	5.2±1.3	2.6±0.3	2.1±0.1	1.9±0.1
8	4000	50	3.2±0.2	2.7±0.1	6.4±0.9	1.6±0.1	1.3±0.05	2.1±0.2	2.0±0.1	1.9±0.1	1.7±0.2	1.5±0.1	1.3±0.05
16	5000	25	2.9±0.1	3.1±0.3	9.4±1.1	1.7±0.1	1.2±0.05	3.7±0.6	2.4±0.1	2.6±0.2	1.7±0.1	1.5±0.1	1.5±0.1

Table 4: Illustration of the demixing of n randomly chosen signals of length m , drawn independently with replacement from Table 3. For COCO and the KMI, we used a Gaussian kernel of size $\sigma = 1$ in the experiments labelled (g), and a Laplace kernel of size $\lambda = 3$ for those experiments labelled (l). In the case of the KCC and KGV, we used $\sigma = 1$ and $\kappa = 2 \times 10^{-2}$ for signals of length $m \leq 1000$, and $\sigma = 0.5$ and $\kappa = 2 \times 10^{-3}$ for the remaining signals. In all cases, we used $\epsilon = 1 \times 10^{-5}$ for the Gaussian kernels, and $\epsilon = 0.01$ for the Laplace kernels. We initialised the kernel methods with Jade in all cases but $n = 16$, for which we used FastICA (due to its more stable output). The performance figures are an average over *Rep.* independent runs. The best results are shown in boldface, as are those results statistically indistinguishable from the best according to a level 0.05 left-tailed paired difference t-test.

n	Fica	Jade	Imax	CFICA	RADICAL	KGV	KMI
2	0.92±0.07	0.99±0.07	1.07±0.10	0.84±0.06	1.02±0.07	0.65±0.05	0.51±0.13
4	0.93±0.03	0.87±0.03	1.09±0.06	0.89±0.03	0.91±0.03	0.62±0.02	0.68±0.03

Table 5: Illustration of the demixing of n music segments of length $m = 55272$, taken from the collection of 17 music samples at (Pearl-mutter). The $n = 2$ case represents an average over 136 samples, and the $n = 4$ case is an average over 120 samples. Details of the KGV and KMI parameters may be found in Section 5.5. The best results are shown in boldface, as those results statistically indistinguishable from the best according to a level 0.05 left-tailed paired difference t-test.

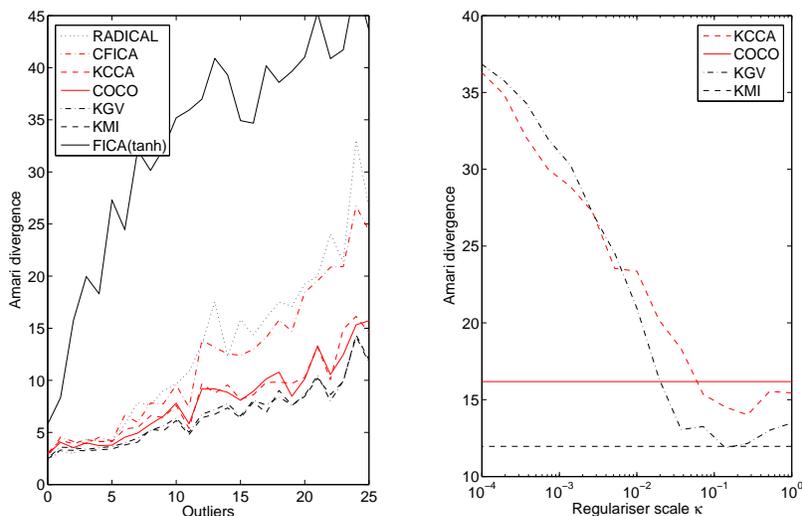


Figure 2: **Left:** Effect of outliers on the performance of the ICA algorithms, for two sources of length $m = 1000$, drawn independently with replacement from Table 3, and corrupted at random observations with outliers at ± 5 (where each sign has probability 0.5). Each point represents an average over 100 independent experiments. The number of corrupted observations in *both* signals is given on the horizontal axis. The kernel methods used $\sigma = 1$, $\varepsilon = 2 \times 10^{-5}$, and $\kappa = 0.11$ (KCC and KGV only). The tanh nonlinearity was used for the FastICA algorithm, since this is more resistant to outliers than the kurtosis (Hyvärinen, 1997). **Right:** Performance of the KCC and KGV as a function of κ for two sources of size $m = 1000$, where 25 outliers were added to each source following the mixing procedure.

5.5 Audio Signal Demixing

Our final experiment involved demixing brief extracts from various musical sources, which were combined using a randomly generated matrix (in the same manner as the artificial signals described in the previous section). A total of 17 different extracts were taken from the ICA benchmark set at (Pearlmutter). These consist of 5 second segments sampled at 11 kHz with a precision of 8 bits, and represent a wide variety of musical genres. While samples of a musical signal are certainly not generated independently and identically in time, many ICA algorithms have nonetheless been applied successfully to this problem, which is why we investigate this benchmark. Indeed, many practical applications of ICA are in a context where complete independence of the unmixed signals is *not* a goal, in theory or in practice: rather, the objective of the linear unmixing is to obtain signals that are relatively “more independent” than the original observations, in the hope that these will be physically interpretable in the light of the system generating the data.

A summary of our results is given in Table 5: the KMI, KGV, and CFICA are statistically indistinguishable for two extracts, and the KGV does best with four extracts, followed by the KMI. In the $n = 2$ case, every possible combination of two different extracts was investigated (for a total of 136 experiments), and the results averaged. We used $\kappa = 2 \times 10^{-3}$, $\sigma = 0.5$, $\varepsilon = 1 \times 10^{-5}$, and a Gaussian kernel for the KGV; and $\lambda = 3$, $\varepsilon = 1 \times 0.01$, and a Laplace kernel for the KMI.

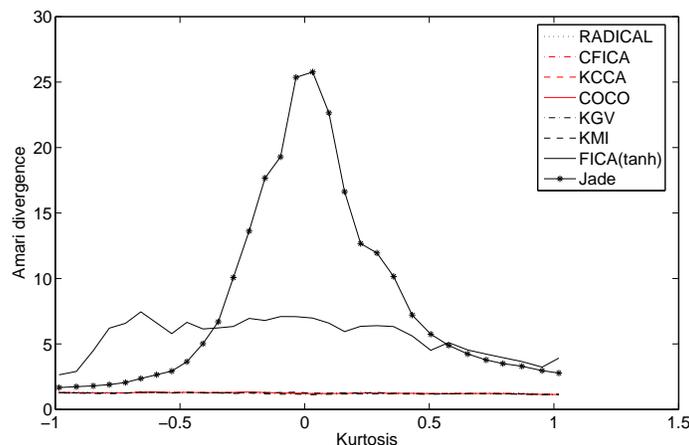


Figure 3: Effect of near-zero kurtosis on the performance of the algorithms, for two signals of length 1000 drawn from a range of mixtures of two Gaussians. Each point represents an average over 100 independent experiments. We used a Gaussian kernel with $\sigma = 1$ and precision $\epsilon = 2 \times 10^{-5}$ for all kernel dependence functionals, and $\kappa = 2 \times 10^{-2}$ for the KCC and KGV.

In both cases, a polishing step was applied to refine the result. For each experiment with $n = 4$, music segments were drawn randomly and without replacement from the 17 available extracts, and the results averaged over 120 repetitions. All kernel algorithm parameters were the same as in the $n = 2$ case besides the Laplace kernel size, which was increased to $\lambda = 4$. In addition, no polishing step was applied to the KGV or KMI, since it caused a drop in performance in both cases.²⁸ Our use of the Laplace kernel in the KMI was motivated by music generally being super-Gaussian (Bell and Sejnowski, 1995). Random permutation of time indices was used to reduce the statistical dependence of adjacent samples in the music, since this was found to improve performance (note that this permutation was carried out on the mixed signals, and was the same for each of the observed mixtures). It is notable that RADICAL, which performs best in the case of noise-free artificial data, does not improve on standard methods in the case of musical sources.

Although the results in Table 5 are quite similar for the KGV and KMI, it is instructive to compare the distribution of the outcomes obtained in each experiment. Generally, the KGV results are more tightly grouped about their mean, whereas the KMI yields more results at smaller Amari divergences, but a larger number of outliers with greater error.

6. Conclusions and Outlook

To conclude this study, we provide a summary of our main results in Section 6.1, and explore directions for future research in Section 6.2.

28. This is perhaps surprising, given that the polishing step caused a minor increase in performance in the $n = 2$ case. On the other hand, the larger dimension of the $n = 4$ problem makes the global minimum harder to find, and diversion to local minima more likely.

6.1 Conclusions

We have introduced two novel functionals to measure independence: the constrained covariance (COCO), which is the spectral norm of the covariance operator between reproducing kernel Hilbert spaces, and the kernel mutual information (KMI), which is a function of the entire spectrum of the empirical estimate of this covariance operator. The first quantity is analogous to the kernel canonical correlation (KCC), which is the spectral norm of the correlation operator; the second is analogous to the kernel generalised variance (KGV), which is a function of the empirical correlation operator spectrum (see Table 1 in the introduction). We prove two main results. First, we describe the class of *all* reproducing kernel Hilbert spaces for which these four functionals determine independence: the RKHSs must be universal. Second, we link the KMI and the KGV with the mutual information, proving the KMI is an upper bound near independence on the Parzen window estimate of the mutual information, and the KGV is a looser upper bound under certain conditions. We emphasise that the KMI and KGV do not require the space partitioning or binning approximations usually associated with estimates of the mutual information (Paninski, 2003).

Our experiments demonstrate the effectiveness of kernel algorithms in ICA, as compared with both standard methods (Jade, Fast ICA, and Extended Infomax); and modern approaches (CFICA, RADICAL). We emphasise that kernel methods (the KMI and KGV in particular) are clearly superior to the alternatives when outlier noise is present in the observations, and are also best at unmixing real (musical) signals. In addition, all modern methods are unaffected by the sources having zero kurtosis, which is not true of earlier algorithms.

Our experiments also point to the superiority of the KMI and KGV over the KCC and COCO in measuring independence. Since independence of two random variables implies that the entire spectrum of the associated covariance (or correlation) operator is zero, it comes as no surprise that measures using the whole spectrum are more robust than those using only the largest singular value. This intuition remains to be formalised, however.

The choice between the KGV and KMI (or, alternatively, COCO and the KCC) is more complicated. The methods proposed by Bach and Jordan (2002a) appear to do well when there is little data available, as in the $n = 2, m = 250$ and $n = 4, m = 1000$ cases in Table 4, although the mechanism by which this is achieved remains unclear. On the other hand, the KCC and KGV do less well when the sample size/number of sources are large. The KGV and KCC can also be more susceptible to noise in the observations, which is particularly apparent when κ becomes small²⁹ (and the bound on mutual information provided by the KGV is looser). Indeed, in our outlier resistance experiments, the KMI and COCO achieve by default the optimal performance of the KCC and KGV with model selection over κ . The absence of a separate regularisation parameter in our kernel functionals therefore greatly simplifies model selection, especially if the observations are known to be corrupted by outliers.

6.2 Directions for Future Study

A number of extensions to this work are readily apparent. For instance, the behaviour of the KMI has not been studied in detail for more than two univariate random variables, besides the discussion in Section 3.2 which guarantees it to be zero when the empirical COCO is zero. In particular, it would be of interest to prove that (30) in Section 3.2 is an upper bound on the Gaussian mutual information, in the manner described in Section 3.1.5 for two random variables. This would incidentally require

29. κ is the regularisation scaling factor for these dependence functionals.

the link between the Gaussian mutual information and the discrete mutual information, described in Section 3.1 for the two variable case, to be extended to a greater number of random variables. The optimisation procedure we use for ICA might also be made faster, for instance by implementing Newton’s method or conjugate gradient descent on the Stiefel manifold (as described by Edelman et al. (1998)), rather than simple gradient descent.

We also need to ensure that both the KMI and COCO approach their population expressions as the sample size increases. In the case of COCO, Gretton et al. (2005b, 2004) give probabilistic bounds for deviations from the expected value using standard tools from uniform convergence theory. The application of these results to the empirical KMI is less clear, however, since the KMI is a *product* of multiple COCO-type quantities, and we do not know what expression it approaches in the population limit. More generally, it is necessary to further investigate methods for model selection (i.e., for choosing the kernel size and type) in COCO and the KMI. It is not presently known whether performance is most effectively tuned by simple cross-validation, using bounds derived from concentration inequalities, or via the properties of Parzen window estimates described by Silverman (1986).

Many real life problems do not fit neatly into the linear ICA framework: we now outline ways in which our kernel dependence functionals might be used to improve performance in these more difficult signal separation problems. First, let us consider the separation of random processes, as opposed to random variables. It is rare in practice to encounter signals that do not depend on their previous outputs. Rather, most real signals exhibit statistical dependencies between the observations at different times (this is obviously true of music, for example). These random processes may be stationary, meaning that their statistical properties (for instance the mean and correlation) do not change over time; or they may be nonstationary. In both cases, however, the time dependence greatly assists in separating signals into independent components, the idea being that the independence of different random processes should hold not only between samples drawn at the same time, but also between samples drawn at *different* times. Approaches to this problem include that of Belouchrani et al. (1997), who separate the signals using decorrelation between the sources at any time shift, and the more general approach of Belouchrani and Amin (1998), who use Cohen’s class time-frequency kernels to transform the signal and facilitate source separation. The former approach is limited since it breaks down when the sources have overlapping spectra, due to its using only a second order dependence measure. Thus, it would be interesting to generalise the approach of Belouchrani et al. (1997) using kernel measures of dependence, rather than correlation. This generalisation has been investigated, using the mutual information as a dependence measure, by Stögbauer et al. (2004).

Another generalisation of ICA is the separation of sources when mixing is nonlinear. This is considerably more difficult than linear ICA, due to the increased complexity of the mixing model. One simplification, which makes the problem more tractable, is the *post-nonlinear* model: the i th component of the observation vector \mathbf{t} is

$$t_i = f_i(\mathbf{b}_i \mathbf{s}), \quad (39)$$

where f_i is the i th (unknown) nonlinearity, and \mathbf{b}_i is the i th row of the mixing matrix \mathbf{B} . This situation corresponds for instance to the observations being distorted by the sensors. Approaches to this problem include the methods of Taleb and Jutten (1999); Achard et al. (2001, 2003)—a comparison of these techniques with COCO and the KMI would therefore be of interest (this would require an efficient optimisation algorithm for our dependence measures under the setting (39)).

Various efforts have also been made to solve the more general case

$$\mathbf{t} = f(\mathbf{s}).$$

This problem requires additional constraints on f , to avoid a trivial solution via the Darmois decomposition (Hyvärinen and Pajunen, 1999) (even then, it is generally the case that each source s_i can only be recovered up to a nonlinear distortion; this is the analogue of the scaling indeterminacy (Theorem 24) in the linear mixing case). It may also be necessary for the observations to arise from random processes, rather than being i.i.d. For instance, according to Hosseini and Jutten (2003), enforcing temporal decorrelation over a single time step is sufficient to test whether the recovered independent processes are simply the result of a Darmois decomposition. While this does not rule out other transforms that return independent signals unrelated to the sources, it suggests that time dependencies have a crucial role to play in general nonlinear mixing. In the scheme suggested by Harmeling et al. (2003), demixing is achieved by mapping the observations to a reproducing kernel Hilbert space, finding a low dimensional basis in the feature space which approximately spans the subspace formed by the observations, and enforcing the second order temporal decorrelation of projections onto this basis. The applicability of the KMI is less clear than in the case of post-nonlinear mixtures, although this might follow from a better understanding of the technique of Harmeling et al. (2003) and its relation to our work.

Finally, Bach and Jordan (2002b) propose using kernel dependence measures in representing probability distributions as tree structured graphical models. Fitting these models requires in particular that the mutual information between pairs of random variables be maximised: thus, Bach and Jordan compare the KGV to a Parzen window estimate of the mutual information in this context. Although the Parzen window approach generally performs better, the KGV is also very effective. We have shown, however, that the KGV is an upper bound (near independence) on the mutual information: thus the KGV performance is a possible indication of the tightness of this upper bound. Given that the KMI is in theory a tighter upper bound than the KGV, it would be interesting to compare its performance with the KGV in this setting.

Acknowledgments

The authors would like to thank to Jean-Yves Audibert and Matthias Seeger, who both discovered errors in our original reasoning for the KMI proof; Francis Bach and Michael Jordan, for providing the kernel ICA code on the web, and for helpful comments; and Aiyu Chen and Erik Learned-Miller, for their assistance in our experimental comparison of the ICA algorithms. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

Appendix A. Proofs

This appendix contains derivations of the main results in the present study, excluding our discussion of the original proofs of Bach and Jordan (2002a) (which are in Appendix B).

A.1 COCO, kernel PCA, and Kernel Target Alignment

In this appendix, we show that COCO is the quantity optimised when obtaining the first principal component in the kernel principal component analysis (kPCA) method of Schölkopf et al. (1998). This can be seen as follows: kPCA satisfies the eigenvalue problem

$$\max_{\|\mathbf{y}\| \leq 1} \mathbf{y}^\top \mathbf{K} \mathbf{y} = \lambda$$

(an inequality is used to keep the constraint set convex). This is rewritten

$$\begin{aligned} \max_{\|\mathbf{y}\| \leq 1} \mathbf{y}^\top \mathbf{K} \mathbf{y} &= \max_{\|\mathbf{y}\| \leq 1} \text{tr}(\mathbf{K} \mathbf{y} \mathbf{y}^\top) \\ &= \max_{\|\mathbf{y}\| \leq 1} \|\mathbf{K} \mathbf{y} \mathbf{y}^\top\|_2, \end{aligned}$$

where the norm in the final line is the largest singular value. The final expression is just $\text{COCO}_{\text{emp}}^2$, with feature space $\mathcal{G} := \mathbb{R}$ and inner product³⁰ $l(y_i, y_j) = y_i y_j$. The difference with respect to the dependence measurement framework described previously is that we now maximise over the members y_i of \mathcal{G} , rather than being given them in advance. This last argument also shows that COCO is optimised in the spectral clustering/kernel target alignment framework of Cristianini et al. (2002).

A.2 Ratio of Determinants

In this appendix, we prove Theorem 11. First, we note that both \mathbf{A} and \mathbf{C} must be positive definite, since they are submatrices of the positive definite matrix (8). Then

$$\begin{aligned} \frac{\left| \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \right|}{|\mathbf{A}| |\mathbf{C}|} &= \frac{\left| \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \right|}{\left| \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \right|} \\ &\stackrel{(a)}{=} \frac{\left| \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \right|}{\left| \begin{bmatrix} \mathbf{A}^{1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{1/2} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{1/2} \end{bmatrix} \right|} \\ &= \left| \begin{bmatrix} \mathbf{A}^{-1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{-1/2} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{-1/2} \end{bmatrix} \right| \\ &= \left| \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1/2} \mathbf{B} \mathbf{C}^{-1/2} \\ \mathbf{C}^{-1/2} \mathbf{B}^\top \mathbf{A}^{-1/2} & \mathbf{I} \end{bmatrix} \right|. \\ &\stackrel{(b)}{=} \left| \mathbf{I} - \mathbf{A}^{-1/2} \mathbf{B} \mathbf{C}^{-1} \mathbf{B}^\top \mathbf{A}^{-1/2} \right| \\ &= \left| \mathbf{I} - \mathbf{C}^{-1/2} \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{B} \mathbf{C}^{-1/2} \right| \\ &\stackrel{(c)}{=} \prod_i (1 - \rho_i^2) \end{aligned}$$

30. Note that the linear kernel used here is *not* universal, and thus COCO is not a general dependence functional in this context: see Section 2.3.

where (a) requires that \mathbf{A} and \mathbf{C} be positive definite,³¹ (b) uses the relation between the determinant of a matrix and that of its Schur complement from Horn and Johnson (1985, p. 22), and (c) uses Theorem 7.3.7 of Horn and Johnson (1985) to determine that ρ_i are the singular values of $\mathbf{A}^{-1/2}\mathbf{B}\mathbf{C}^{-1/2}$. Note that since (8) has only positive eigenvalues, and the determinant of a symmetric matrix is the product of the eigenvalues, we are guaranteed

$$\prod_i (1 - \rho_i^2) > 0.$$

From Horn and Johnson (1985, Theorem 7.3.7), we can write ρ_i as the positive solutions of the eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{A}^{-1/2}\mathbf{B}\mathbf{C}^{-1/2} \\ \mathbf{C}^{-1/2}\mathbf{B}^\top\mathbf{A}^{-1/2} & \mathbf{0} \end{bmatrix} \mathbf{b}_i = \rho_i \mathbf{b}_i,$$

bearing in mind that these solutions come in pairs with equal magnitude and opposite sign. Rearranging and making an appropriate change of variables yields the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{0} \end{bmatrix} \mathbf{a}_i = \rho_i \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \mathbf{a}_i.$$

A.3 Determinant Form of the Gaussian Mutual Information

In this section, we give a derivation of (18) in Section 3.1.3, which states that

$$I(\mathbf{x}_G; \mathbf{y}_G) = -\frac{1}{2} \log \left(\left| \mathbf{I}_y - (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top)^\top \mathbf{D}_x^{-1} (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top) \mathbf{D}_y^{-1} \right| \right). \quad (40)$$

This result was given without proof by Bach and Jordan (2002a, Appendix B). We begin with the mutual information between \mathbf{x}_G and \mathbf{y}_G , which is written

$$I(\mathbf{x}_G; \mathbf{y}_G) = -\frac{1}{2} \log \left(\prod_i (1 - \rho_i^2) \right), \quad (41)$$

where ρ_i are the positive solutions to the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top \\ (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top)^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} \mathbf{D}_x - \mathbf{p}_x \mathbf{p}_x^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_y - \mathbf{p}_y \mathbf{p}_y^\top \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} \quad (42)$$

(this can be found by substituting the covariances (15)-(17) into (10)). Note that both $\mathbf{D}_x - \mathbf{p}_x \mathbf{p}_x^\top$ and $\mathbf{D}_y - \mathbf{p}_y \mathbf{p}_y^\top$ have rank $l_x - 1$ and $l_y - 1$ respectively, and are not invertible.³² To see this, we make the expansions

$$\begin{aligned} \mathbf{D}_x - \mathbf{p}_x \mathbf{p}_x^\top &= \mathbf{D}_x (\mathbf{I}_{l_x} - \mathbf{1}_{l_x} \mathbf{p}_x^\top) = \mathbf{D}_x \mathbf{E}_x, \\ \mathbf{D}_y - \mathbf{p}_y \mathbf{p}_y^\top &= \mathbf{D}_y (\mathbf{I}_{l_y} - \mathbf{1}_{l_y} \mathbf{p}_y^\top) = \mathbf{D}_y \mathbf{E}_y, \end{aligned}$$

31. A matrix has a square root if and only if it is positive definite.

32. This is why we use (41) as our expression for the mutual information, rather than the ratio of determinants (7) (which would be undefined here).

where $\mathbf{E}_x := \mathbf{I}_x - \mathbf{1}_{l_x} \mathbf{p}_x^\top$ and $\mathbf{E}_y := \mathbf{I}_y - \mathbf{1}_{l_y} \mathbf{p}_y^\top$ have zero eigenvalues corresponding to the eigenvectors $\frac{1}{\sqrt{l_x}} \mathbf{1}_{l_x}$ and $\frac{1}{\sqrt{l_y}} \mathbf{1}_{l_y}$, respectively. In addition, we note that

$$\begin{aligned} (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top) \mathbf{E}_y &= (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top) (\mathbf{I}_y - \mathbf{1}_{l_y} \mathbf{p}_y^\top) \\ &= \mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top - \mathbf{P}_{xy} \mathbf{1}_{l_y} \mathbf{p}_y^\top + \mathbf{p}_x \mathbf{p}_y^\top \mathbf{1}_{l_y} \mathbf{p}_y^\top \\ &= \mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top - \mathbf{p}_x \mathbf{p}_y^\top + \mathbf{p}_x \mathbf{p}_y^\top \\ &= \mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top, \end{aligned}$$

with an analogous result for $(\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top)^\top \mathbf{E}_x$. We may therefore write (42) as

$$\begin{bmatrix} \mathbf{0} & (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top) \mathbf{E}_y \\ (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top)^\top \mathbf{E}_x & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} \mathbf{D}_x \mathbf{E}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_y \mathbf{E}_y \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix},$$

from which we obtain a generalised eigenvalue problem with identical eigenvalues ρ_i ,

$$\begin{bmatrix} \mathbf{0} & \mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top \\ (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top)^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{e}_i \\ \mathbf{f}_i \end{bmatrix} = \rho_i \begin{bmatrix} \mathbf{D}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_y \end{bmatrix} \begin{bmatrix} \mathbf{e}_i \\ \mathbf{f}_i \end{bmatrix}.$$

Since \mathbf{D}_x and \mathbf{D}_y have full rank, we may now apply Theorem 11 to obtain (40).

A.4 Details of Definition 13

In this section, we derive the Parzen window estimate of the Gaussian mutual information provided in Definition 13. The kernel density (Parzen window) estimates for $\mathbf{p}_{x,y}$ and its marginals, on the basis of the sample \mathbf{z} , are

$$\begin{aligned} \hat{\mathbf{p}}_x(x) &= \frac{1}{m} \sum_{l=1}^m \kappa(x_l - x), & \hat{\mathbf{p}}_y(y) &= \frac{1}{m} \sum_{l=1}^m \kappa(y_l - y), \\ \hat{\mathbf{p}}_{x,y}(x,y) &= \frac{1}{m} \sum_{l=1}^m \kappa(x_l - x) \kappa(y_l - y), \end{aligned}$$

where the kernel argument indicates which kernel is used, to simplify notation. We require approximations to the terms in the Gaussian mutual information, as described in (18). We therefore define the vectors $\hat{\mathbf{p}}_x, \hat{\mathbf{p}}_y$, and the matrix $\hat{\mathbf{P}}_{xy}$, using the expectations in (12)-(14) computed with these kernel expressions;

$$\hat{\mathbf{E}}_{x,y}(\check{\mathbf{x}} \check{\mathbf{y}}^\top) = \hat{\mathbf{P}}_{xy}, \quad (43)$$

$$\hat{\mathbf{E}}_x(\check{\mathbf{x}}) = \hat{\mathbf{p}}_x, \quad (44)$$

$$\hat{\mathbf{E}}_x(\check{\mathbf{x}} \check{\mathbf{x}}^\top) = \hat{\mathbf{D}}_x. \quad (45)$$

In the limit where Δ_x, Δ_y are small (and thus, by implication, $l_x \gg m, l_y \gg m, \sigma_x \gg \Delta_x$, and $\sigma_y \gg \Delta_y$, where σ_x and σ_y define the kernel sizes), we make the approximations

$$\hat{\mathbf{E}}_x((\check{\mathbf{x}})_i) = \hat{\mathbf{P}}_{\check{x}}(i) = \frac{1}{m} \int_{q_i}^{q_i + \Delta_x} \sum_{l=1}^m \kappa(x_l - x) dx \approx \frac{\Delta_x}{m} \sum_{l=1}^m \kappa(x_l - q_i),$$

$$\widehat{\mathbf{E}}_x \left(\left(\check{\mathbf{x}} \check{\mathbf{x}}^\top \right)_{i,j} \right) \approx \begin{cases} \frac{\Delta_x}{m} \sum_{l=1}^m \kappa(x_l - q_i) & i = j \\ 0 & \text{otherwise} \end{cases},$$

and

$$\begin{aligned} \widehat{\mathbf{E}}_{x,y} \left(\left(\check{\mathbf{x}} \check{\mathbf{y}}^\top \right)_{i,j} \right) &= \widehat{\mathbf{P}}_{\check{x},\check{y}}(i,j) = \frac{1}{m} \int_{q_i}^{q_i+\Delta_x} \int_{r_j}^{r_j+\Delta_y} \sum_{l=1}^m \kappa(x_l - x) \kappa(y_l - y) dx dy \\ &\approx \frac{\Delta_x \Delta_y}{m} \sum_{l=1}^m \kappa(x_l - q_i) \kappa(y_l - r_j). \end{aligned}$$

Before proceeding further, we define two matrices of kernel inner products to simplify our notation. Namely,

$$\mathbf{K}_l := \begin{bmatrix} \kappa(q_1 - x_1) & \dots & \kappa(q_1 - x_m) \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \kappa(q_{l_x} - x_1) & \dots & \kappa(q_{l_x} - x_m) \end{bmatrix}, \quad \mathbf{L}_l := \begin{bmatrix} \kappa(r_1 - y_1) & \dots & \kappa(r_1 - y_m) \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \kappa(r_{l_y} - y_1) & \dots & \kappa(r_{l_y} - y_m) \end{bmatrix}, \quad (46)$$

where we write the above in such a manner as to indicate $l_x \gg m$ and $l_y \gg m$. We now use the above results to re-write (43)-(45) as respectively

$$\widehat{\mathbf{P}}_{xy} - \widehat{\mathbf{p}}_x \widehat{\mathbf{p}}_y^\top \approx \frac{\Delta_x \Delta_y}{m} \left(\mathbf{K}_l \mathbf{L}_l^\top - \frac{1}{m} \mathbf{K}_l \mathbf{1}_m \mathbf{1}_m^\top \mathbf{L}_l^\top \right) = \frac{\Delta_x \Delta_y}{m} \mathbf{K}_l \mathbf{H} \mathbf{L}_l^\top,$$

$$\widehat{\mathbf{D}}_x \approx \frac{\Delta_x}{m} \text{diag}(\mathbf{K}_l \mathbf{1}_m) =: \frac{\Delta_x^2}{m} \mathbf{D}_l^{(x)},$$

and

$$\widehat{\mathbf{D}}_y \approx \frac{\Delta_y}{m} \text{diag}(\mathbf{L}_l \mathbf{1}_m) =: \frac{\Delta_y^2}{m} \mathbf{D}_l^{(y)},$$

where we introduce the terms

$$\mathbf{D}_l^{(x)} = \frac{1}{\Delta_x} \begin{bmatrix} \sum_{l=1}^m \kappa(q_1 - x_l) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sum_{l=1}^m \kappa(q_{l_x} - x_l) \end{bmatrix} \quad (47)$$

and

$$\mathbf{D}_l^{(y)} = \frac{1}{\Delta_y} \begin{bmatrix} \sum_{l=1}^m \kappa(r_1 - y_l) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sum_{l=1}^m \kappa(r_{l_y} - y_l) \end{bmatrix}. \quad (48)$$

With these substitutions, we can rewrite

$$\left(\widehat{\mathbf{D}}_x \right)^{-1/2} \left(\widehat{\mathbf{P}}_{xy} - \widehat{\mathbf{p}}_x \widehat{\mathbf{p}}_y^\top \right) \left(\widehat{\mathbf{D}}_y \right)^{-1/2} \approx \left(\mathbf{D}_l^{(x)} \right)^{-1/2} \left(\mathbf{K}_l \mathbf{H} (\mathbf{L}_l)^\top \right) \left(\mathbf{D}_l^{(y)} \right)^{-1/2},$$

which results in our definition.

A.5 Proof of Theorem 16

Our proof of Theorem 16 requires the following lemma.

Lemma 27 (Singular values of a matrix product) *Let \mathbf{A} , \mathbf{B} be $m \times n$ matrices, $q := \min(m, n)$, and \mathbf{A} have singular values $\sigma_1(\mathbf{A}), \dots, \sigma_q(\mathbf{A})$ (ordered from largest to smallest). Then $\sigma_1(\mathbf{AB}^\top) \leq \sigma_1(\mathbf{A})\sigma_1(\mathbf{B})$ and*

$$\sigma_q(\mathbf{AB}^\top) \leq \min\{\sigma_q(\mathbf{A})\sigma_1(\mathbf{B}), \sigma_1(\mathbf{A})\sigma_q(\mathbf{B})\}.$$

This is a special case of a result of Horn and Johnson (1985, p. 423). We now proceed with the proof. The principle we will follow is straightforward: we want to upper bound the Gaussian mutual information in (20) by upper bounding *each* of the $\hat{\rho}_i$ that define it. Indeed, if we can find a matrix to replace (21) with singular values $\alpha_i \geq \hat{\rho}_i$ for all i , it follows that $-\frac{1}{2} \log(\prod_i (1 - \alpha_i^2)) \geq -\frac{1}{2} \log(\prod_i (1 - \hat{\rho}_i^2))$. First, we note that $\pm \hat{\rho}_i$ are the eigenvalues of the matrix

$$\underbrace{\begin{bmatrix} (\mathbf{D}_l^{(x)})^{-1} & \mathbf{0} \\ \mathbf{0} & (\mathbf{D}_l^{(y)})^{-1} \end{bmatrix}}_{\mathbf{D}^{-1}} \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{K}_l \mathbf{H}(\mathbf{L}_l)^\top \\ \mathbf{L}_l \mathbf{H}(\mathbf{K}_l)^\top & \mathbf{0} \end{bmatrix}}_{\mathbf{E}}.$$

According to (22), $\mathbf{D}_l^{(x)}$ is a diagonal matrix with j th entry $\frac{1}{\Delta_x} \sum_{i=1}^m \kappa(x_i - q_j)$, which is an unnormalised Parzen window estimate of \mathbf{p}_x at grid point q_j (an analogous result holds for $\mathbf{D}_l^{(y)}$). It follows that \mathbf{D} is diagonal, and we denote its i th largest value as d_i (i.e., d_1 is the overall maximum); we also define σ_i to be the i th singular value of \mathbf{E} . We may obtain a new matrix with singular values $\alpha_i \geq \hat{\rho}_i$ by replacing the diagonal entries of \mathbf{D} with their smallest value,³³

$$\begin{aligned} \mathbf{D} &\rightarrow \min_i (d_i) \mathbf{I} \\ &= \frac{v_z}{\Delta} \mathbf{I}, \end{aligned} \tag{49}$$

where $v_z = \min\{v_x, v_y\}$ and

$$v_x := \min_{j \in \{1 \dots l_x\}} \sum_{i=1}^m \kappa(x_i - q_j), \quad v_y := \min_{j \in \{1 \dots l_y\}} \sum_{i=1}^m \kappa(y_i - r_j). \tag{50}$$

The singular values α_i of $(\frac{v_z}{\Delta} \mathbf{I})^{-1} \mathbf{E}$ satisfy³⁴

$$\begin{aligned} \hat{\rho}_i &\leq \min\{d_{l_x+l_y}^{-1} \sigma_i, d_{l_x+l_y-i+1}^{-1} \sigma_1\} \\ &\leq d_{l_x+l_y}^{-1} \sigma_i \\ &= \frac{\Delta}{v_z} \sigma_i = \alpha_i \end{aligned}$$

33. We assume without loss of generality that $\Delta_x = \Delta_y = \Delta$, since this simplifies notation.

34. Bear in mind that due to the ordering of the singular values, $\max_j d_j^{-1} = d_{l_x+l_y}^{-1}$; and the d_j^{-1} are sorted in reverse order to the d_j .

for all i , where the first inequality derives from Lemma 27. Rather than computing the minima in (50) over the grid, however, we may simply use

$$v_{\mathbf{x}} := \min_{j \in \{1 \dots m\}} \sum_{i=1}^m \kappa(x_i - x_j), \quad v_{\mathbf{y}} := \min_{j \in \{1 \dots m\}} \sum_{i=1}^m \kappa(y_i - y_j),$$

which are respectively the smallest (unnormalised) Parzen window estimates of $\mathbf{p}_{\mathbf{x}}$ and $\mathbf{p}_{\mathbf{y}}$ at any *sample point*: these approach the smallest values of $\mathbf{p}_{\mathbf{x}}$ on \mathcal{X} , and of $\mathbf{p}_{\mathbf{y}}$ on \mathcal{Y} , as the sample size increases (the densities are bounded away from zero by assumption).

Having made the replacement in (49), it is straightforward to take a limit in which the grid becomes infinitely fine. We begin by rearranging the Lemma 13 definition as

$$\begin{aligned} \widehat{l}(\hat{\mathbf{x}}; \hat{\mathbf{y}}) &\leq -\frac{1}{2} \log \left| \mathbf{I} - \left(\frac{\Delta}{v_{\mathbf{z}}} \right)^2 \left(\mathbf{K}_l \mathbf{H} (\mathbf{L}_l)^\top \right) \left(\mathbf{K}_l \mathbf{H} (\mathbf{L}_l)^\top \right)^\top \right| \\ &= -\frac{1}{2} \log \left| \mathbf{I} - \left(\frac{\Delta}{v_{\mathbf{z}}} \right)^2 \left(\mathbf{H} \mathbf{K}_l^\top \mathbf{K}_l \mathbf{H} \right) \left(\mathbf{H} \mathbf{L}_l^\top \mathbf{L}_l \mathbf{H} \right) \right|. \end{aligned}$$

We then have the limiting result

$$\begin{aligned} \lim_{l_x \rightarrow \infty} \left(\frac{\Delta_x}{v_{\mathbf{z}}} \mathbf{K}_l^\top \mathbf{K}_l \right)_{i,j} &= v_{\mathbf{z}}^{-1} \lim_{l_x \rightarrow \infty} \Delta_x \sum_{p=1}^{l_x} \kappa(x_i - q_p) \kappa(x_j - q_p) \\ &= v_{\mathbf{z}}^{-1} \int_{\mathcal{X}} \kappa(x_i - q) \kappa(x_j - q) dq \\ &= v_{\mathbf{z}}^{-1} k(x_i, x_j), \end{aligned}$$

where we recover our RKHS kernel as the convolution of the kernel density functions at each pair of data points.

A.6 Proof of Theorem 18

In this section, we prove that the KGV upper bounds the KMI when conditions (28) hold. We recall the definition of the *unregularised* KGV,³⁵ which occurs at $\theta = 1$. It follows from Lemma 9 that

$$\text{KGV}(\mathbf{z}; F, G, 1) = \infty,$$

since the associated eigenvalues ρ_i in (27) are all either 1, -1 , or 0 (given we use universal kernels, there will be at least one pair of non-zero eigenvalues). Conversely, when $\theta = 0$, we recover the KMI. It remains to show that increasing θ from 0 to 1 causes the KGV to increase monotonically.

We may rearrange the eigenvalue problem in (27) as

$$\begin{bmatrix} \mathbf{I} & \left(\theta \tilde{\mathbf{K}} + (1 - \theta) v I \right)^{-1} \tilde{\mathbf{L}} \\ \left(\theta \tilde{\mathbf{L}} + (1 - \theta) v I \right)^{-1} \tilde{\mathbf{K}} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = (1 + \rho_i) \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix}.$$

35. We emphasise that only the regularised KGV is used in practice.

Then

$$\begin{aligned} \text{KGV}(\mathbf{z}; F, G, \theta) &= -\log \left| \begin{bmatrix} \mathbf{I} & (\theta \tilde{\mathbf{K}} + (1-\theta)\mathbf{v}_z \mathbf{I})^{-1} \tilde{\mathbf{L}} \\ (\theta \tilde{\mathbf{L}} + (1-\theta)\mathbf{v}_z \mathbf{I})^{-1} \tilde{\mathbf{K}} & \mathbf{I} \end{bmatrix} \right| \\ &= -\log \left| \mathbf{I} - (\theta \tilde{\mathbf{K}} + (1-\theta)\mathbf{v}_z \mathbf{I})^{-1} \tilde{\mathbf{L}} (\theta \tilde{\mathbf{L}} + (1-\theta)\mathbf{v}_z \mathbf{I})^{-1} \tilde{\mathbf{K}} \right|. \end{aligned}$$

We now use the result that if $\mathbf{A}' \succ \mathbf{A} \succ \mathbf{0}$ and $\mathbf{B}' \succ \mathbf{B} \succ \mathbf{0}$, then $\mathbf{A}'\mathbf{B}' \succ \mathbf{A}\mathbf{B}$ (this is a straightforward corollary to Theorem 7.7.3 of Horn and Johnson, 1985). The desired result then holds as long as

$$\theta' \tilde{\mathbf{K}} + (1-\theta')\mathbf{v}_z \mathbf{I} \prec \theta \tilde{\mathbf{K}} + (1-\theta)\mathbf{v}_z \mathbf{I}$$

when $\theta' > \theta$ (as well as the analogous result for $\theta \tilde{\mathbf{L}} + (1-\theta)\mathbf{v}_z \mathbf{I}$), which means

$$(\theta - \theta') \tilde{\mathbf{K}} + (\theta' - \theta)\mathbf{v}_z \mathbf{I} \succ 0 \quad \text{and} \quad (\theta - \theta') \tilde{\mathbf{L}} + (\theta' - \theta)\mathbf{v}_z \mathbf{I} \succ 0,$$

or

$$\mathbf{v}_z \mathbf{I} - \tilde{\mathbf{K}} \succ 0 \quad \text{and} \quad \mathbf{v}_z \mathbf{I} - \tilde{\mathbf{L}} \succ 0. \quad (51)$$

A.7 Proof of Lemma 22

In this section, we show that the multivariate KMI is zero if and only if the empirical COCO between each pair of random variables is zero. This may be shown via a minor adaptation of the corresponding proof of Bach and Jordan (2002a, Appendix A.2). First, we may rewrite each factor $\check{\lambda}_j + 1$ in (30) as the solution to

$$\begin{bmatrix} \mathbf{I} & \mathbf{v}_z^{-1} \tilde{\mathbf{K}}_1^{1/2} \tilde{\mathbf{K}}_2^{1/2} & \dots & \mathbf{v}_z^{-1} \tilde{\mathbf{K}}_1^{1/2} \tilde{\mathbf{K}}_n^{1/2} \\ \mathbf{v}_z^{-1} \tilde{\mathbf{K}}_2^{1/2} \tilde{\mathbf{K}}_1^{1/2} & \mathbf{I} & \dots & \mathbf{v}_z^{-1} \tilde{\mathbf{K}}_2^{1/2} \tilde{\mathbf{K}}_n^{1/2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_z^{-1} \tilde{\mathbf{K}}_n^{1/2} \tilde{\mathbf{K}}_1^{1/2} & \mathbf{v}_z^{-1} \tilde{\mathbf{K}}_n^{1/2} \tilde{\mathbf{K}}_2^{1/2} & \dots & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{d}_{1,j} \\ \mathbf{d}_{2,j} \\ \vdots \\ \mathbf{d}_{n,j} \end{bmatrix} = (\check{\lambda}_j + 1) \begin{bmatrix} \mathbf{d}_{1,j} \\ \mathbf{d}_{2,j} \\ \vdots \\ \mathbf{d}_{n,j} \end{bmatrix},$$

where $\tilde{\mathbf{K}}_i^{1/2} \mathbf{c}_{i,j} = \mathbf{d}_{i,j}$, bearing in mind that the determinant of the left hand matrix is the product of these eigenvalues. Since the left hand matrix is symmetric, the trace is equal to the sum of the eigenvalues, and

$$\sum_{j=1}^{mn} (\check{\lambda}_j + 1) = mn. \quad (52)$$

Assuming without loss of generality that the the mn th eigenvalue corresponds to $\check{\lambda}_{\max} := \lambda_{\max}/\nu_{\mathbf{z}}$, we rewrite (30) as

$$\begin{aligned}
 -\frac{1}{2} \log \prod_{j=1}^{mn} (1 + \check{\lambda}_j) &= -\frac{1}{2} \log(1 + \check{\lambda}_{\max}) - \frac{1}{2} \log \prod_{j=1}^{mn-1} (1 + \check{\lambda}_j) \\
 &= -\frac{1}{2} \log(1 + \check{\lambda}_{\max}) - \frac{mn-1}{2} \sum_{j=1}^{mn-1} \frac{1}{mn-1} \log(1 + \check{\lambda}_j) \\
 &\geq -\frac{1}{2} \log(1 + \check{\lambda}_{\max}) - \frac{mn-1}{2} \log \left(\frac{1}{mn-1} \sum_{j=1}^{mn-1} (1 + \check{\lambda}_j) \right) \\
 &= -\frac{1}{2} \log(1 + \check{\lambda}_{\max}) - \frac{mn-1}{2} \log \left(\frac{mn - \check{\lambda}_{\max} - 1}{mn-1} \right),
 \end{aligned}$$

where the penultimate line uses Jensen’s inequality, and we substitute (52) in the final line. The resulting expression is strictly convex with respect to $\check{\lambda}_{\max}$ (its second derivative is everywhere positive), and has a global minimum at $\check{\lambda}_{\max} = 0$. It follows that (30) is likewise minimised at $\text{KMI}(\mathbf{z}; \mathcal{F}_{X_1}, \dots, \mathcal{F}_{X_n}) = 0$ (at which point $\check{\lambda}_j = 0$ for all j), and that this corresponds to the point at which all pairs of empirical constrained covariances are zero, using Definition 19 and Lemma 20.

Appendix B. Discussion of Bach and Jordan’s Derivation of the KGV

This appendix contains a demonstration of the need for regularisation when estimating the canonical correlation in high dimensional spaces, and a discussion of the original KGV derivation of Bach and Jordan (2002a).

B.1 Computation of the Unregularised Kernel Canonical Correlations

In this section, we prove Lemma 9, which is used to show a regularised empirical estimate for the kernel canonical correlates is needed when the associated RKHSs have high dimension. We begin with (5), which we restate below for reference;

$$\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ \tilde{\mathbf{L}}\tilde{\mathbf{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} (\tilde{\mathbf{K}})^2 & \mathbf{0} \\ \mathbf{0} & (\tilde{\mathbf{L}})^2 \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix}.$$

This is equivalent to

$$\begin{bmatrix} \mathbf{0} & (\tilde{\mathbf{K}}^-)^2 \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ (\tilde{\mathbf{L}}^-)^2 \tilde{\mathbf{L}}\tilde{\mathbf{K}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix},$$

where we use the pseudoinverses since the Gram matrices do not have full rank. If we recall that \mathbf{H} is the centring matrix, then the solutions ρ_i correspond to the solutions of

$$\begin{aligned}
 0 &= \left| \begin{array}{cc} -\rho\mathbf{I} & (\tilde{\mathbf{K}}^-)^2 \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ (\tilde{\mathbf{L}}^-)^2 \tilde{\mathbf{L}}\tilde{\mathbf{K}} & -\rho\mathbf{I} \end{array} \right| \\
 &= |\rho\mathbf{I}| \left| \rho\mathbf{I} - \frac{1}{\rho} (\tilde{\mathbf{L}}^-)^2 \tilde{\mathbf{L}}\tilde{\mathbf{K}} (\tilde{\mathbf{K}}^-)^2 \tilde{\mathbf{K}}\tilde{\mathbf{L}} \right| \\
 &= |\rho\mathbf{I}| \left| \rho\mathbf{I} - \frac{1}{\rho} \mathbf{H} \right| \\
 &= \rho^m \frac{(\rho^2 - 1)^{m-1}}{\rho^{m-2}},
 \end{aligned}$$

which has $m - 1$ roots $+1$, $m - 1$ roots -1 , and 2 roots 0. To avoid this problem, a regularised empirical estimate is used, as shown by Bach and Jordan (2002a); Fukumizu et al. (2005); Leurgans et al. (1993).

B.2 Discussion of the KGV Proof of Bach and Jordan (2002a)

In this section, we describe a possible problem in the derivation by Bach and Jordan (2002a, Appendix B) of the kernel generalised variance (KGV). We begin with a quick summary of the steps from Section 3 needed to get us to the point where the proof begins.³⁶ Assume that \mathcal{X} and \mathcal{Y} are both bounded intervals on \mathbb{R} . In Section 3.1.2, we recall the standard result from Cover and Thomas (1991) that the mutual information $I(x, y)$ between two real-valued, univariate random variables $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ can be approximated by imposing a uniform grid of size $l_x \times l_y$ over $\mathcal{X} \times \mathcal{Y}$, and defining a multinomial distribution over the discrete valued random variables $\hat{x} \in \{1, \dots, l_x\}$ and $\hat{y} \in \{1, \dots, l_y\}$ using the probability mass in the resulting bins (this multinomial distribution is described by the matrix \mathbf{P}_{xy} of joint probabilities, with marginal distribution vectors \mathbf{p}_x and \mathbf{p}_y).³⁷ We denote the resulting discrete mutual information as $I(\hat{x}; \hat{y})$. In Section 3.1.3, we approximate $I(\hat{x}; \hat{y})$ using the *Gaussian* mutual information $I(\mathbf{x}_G; \mathbf{y}_G)$ between vectors $\mathbf{x}_G; \mathbf{y}_G$, defined to have the same covariance as $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, where $\hat{x} = i$ is equivalent to $(\hat{\mathbf{x}})_i = 1$ and $(\hat{\mathbf{x}})_{j:j \neq i} = 0$ (likewise for \hat{y}). Bach and Jordan (2002a, Appendix B.1) show this approximation holds when the random variables are close to independence, in which case

$$I(\hat{x}; \hat{y}) \approx I(\mathbf{x}_G; \mathbf{y}_G) = -\frac{1}{2} \log \left(\prod_i (1 - \rho_i^2) \right),$$

where ρ_i are the positive solutions to the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top \\ (\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top)^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} \mathbf{D}_x - \mathbf{p}_x \mathbf{p}_x^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_y - \mathbf{p}_y \mathbf{p}_y^\top \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix},$$

and $\mathbf{D}_x = \text{diag}(\mathbf{p}_x)$, $\mathbf{D}_y = \text{diag}(\mathbf{p}_y)$ (see (41) in Appendix A.3).

36. The reader is strongly advised to consult Sections 3.1.1-3.1.3 before proceeding, since the following discussion might not otherwise make much sense.

37. The approximation becomes exact in the limit of an infinitely fine grid.

We are now at the point where we can describe the reasoning of Bach and Jordan (2002a, Appendix B.3) in establishing a link between $I(\hat{\mathbf{x}}; \hat{\mathbf{y}})$ and the KGV. Rather than replacing $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ by \mathbf{x}_G and \mathbf{y}_G , we may instead replace them with the *smoothed approximations*

$$\mathbf{k}_l = \Delta_x [k(x, q_1) \ \cdots \ k(x, q_{l_x})]^\top \quad \text{and} \quad \mathbf{l}_l = \Delta_y [l(y, r_1) \ \cdots \ l(y, r_{l_y})]^\top \quad (53)$$

to \mathbf{x}_G and \mathbf{y}_G , respectively, where $k(\cdot, \cdot)$ and $l(\cdot, \cdot)$ are the RKHS kernels for \mathcal{F} and \mathcal{G} , and the grid coordinates $\mathbf{q} := (q_1, \dots, q_{l_x})$ and $\mathbf{r} := (r_1, \dots, r_{l_y})$ are defined in Section 3.1.2.³⁸ We can of course specify the Gaussian mutual information $I(\mathbf{k}_l; \mathbf{l}_l)$ between these smoothed vectors, using the appropriate log ratio of determinants. Two questions then arise. First, does this smoothed approximation $I(\mathbf{k}_l; \mathbf{l}_l)$ approach the Gaussian mutual information $I(\mathbf{x}_G; \mathbf{y}_G)$ as the kernel size drops? Second, under what conditions does the empirical estimate of $I(\mathbf{k}_l; \mathbf{l}_l)$ correspond to the KGV? We now describe the approach of Bach and Jordan (2002a) to solving the first question, and postpone discussion of the second question to the end of the section.

The link between the Gaussian approximation to the discrete mutual information and the KGV could be shown by demonstrating

$$\mathbf{P}_{xy} \stackrel{?}{\approx} \Delta_x \Delta_y \mathbf{E}_{x,y} (\mathbf{k}_l \mathbf{l}_l^\top), \quad \mathbf{D}_x \stackrel{?}{\approx} \Delta_x^2 \mathbf{E}_x (\mathbf{k}_l \mathbf{k}_l^\top), \quad \mathbf{p}_x \stackrel{?}{\approx} \Delta_x \mathbf{E}_x (\mathbf{k}_l) \quad (54)$$

under appropriate conditions, with similar results for the terms in y . We consider the case where both kernels are Gaussian; that is,

$$k(x - q_i) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{(x - q_i)^2}{2\sigma_x^2}\right),$$

$$l(y - r_j) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(y - r_j)^2}{2\sigma_y^2}\right),$$

bearing in mind that the impulse function is a limiting case (Bracewell, 1986);

$$\delta_{q_i}(x) = \lim_{\sigma_x \rightarrow 0} \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{(x - q_i)^2}{2\sigma_x^2}\right) := \lim_{\sigma_x \rightarrow 0} k(x - q_i). \quad (55)$$

To compute the covariance structure of the vectors in (53), we require expressions for the expectations

$$\mathbf{E}_{x,y} (\mathbf{k}_l \mathbf{l}_l^\top), \quad \mathbf{E}_x (\mathbf{k}_l), \quad \mathbf{E}_x (\mathbf{k}_l \mathbf{k}_l^\top),$$

$$\mathbf{E}_y (\mathbf{l}_l \mathbf{l}_l^\top), \quad \mathbf{E}_y (\mathbf{l}_l).$$

The expectation of individual entries in the matrix $\mathbf{k}_l \mathbf{l}_l^\top$ is

$$\mathbf{E}_{x,y} [k(q_i, x)l(r_j, y)] = \int_x \int_y k(x - q_i)l(y - r_j) \mathbf{p}_{x,y}(x, y) dx dy$$

$$= [k(x)l(y) \star \mathbf{p}_{x,y}(x, y)](q_i, r_j),$$

38. We use a sans-serif font to define \mathbf{k}_l and \mathbf{l}_l , to indicate that these are random vectors. In addition, Bach and Jordan (2002a) define these quantities without multiplying by Δ_x and Δ_y , but we believe these scalings to be necessary: see below.

which is the convolution of the product of kernels with the underlying (unknown) density $\mathbf{p}_{x,y}(x,y)$ of the random variables x,y , evaluated at q_i, r_j . Since the kernels are normalised, the above expectation is also a probability density, smoothed by $k(x)l(y)$. Similarly,

$$\begin{aligned} \mathbf{E}_x [k(q_i, x)k(q_j, x)] &= \int_x k(x - q_i)k(x - q_j)\mathbf{p}_x(x)dx \\ &\approx \begin{cases} [k^2(x) \star \mathbf{p}_x(x)](q_i) & i = j \\ 0 & \text{otherwise} \end{cases}, \end{aligned}$$

where the above assumes $\sigma_x \ll \Delta_x \ll 1$. Note, however, that

$$k^2(x - q_i) = \frac{1}{2\pi\sigma_x^2} \exp\left(-\frac{(x - q_i)^2}{\sigma_x^2}\right) \quad (56)$$

$$= \frac{1}{2\sigma_x\sqrt{\pi}} \times \frac{1}{\sqrt{\pi\sigma_x^2}} \exp\left(-\frac{(x - q_i)^2}{\sigma_x^2}\right), \quad (57)$$

and thus $k^2(x)$ is *not* a probability density (the integral over \mathbb{R} is equal to $\frac{1}{2\sigma_x\sqrt{\pi}}$). Finally,

$$\begin{aligned} \mathbf{E}_x [k(q_i, x)] &= \int_{\mathbb{R}} k(x - q_i)\mathbf{p}_x(x)dx \\ &= [k(x) \star \mathbf{p}_x(x)](q_i). \end{aligned}$$

In the light of these observations, it might seem that the relations in (54) ought to hold in the limit as $\Delta_x, \Delta_y \rightarrow 0$ and $\sigma_x, \sigma_y \rightarrow 0$, so long as $\sigma_x \ll \Delta_x$ and $\sigma_y \ll \Delta_y$: the grid size must be small to allow us to make the approximations

$$\mathbf{P}_{\hat{x}}(i) = \int_{q_i}^{q_i + \Delta_x} \mathbf{p}_x(x) dx \approx \Delta_x \mathbf{p}_x(q_i)$$

and

$$\mathbf{P}_{\hat{x}, \hat{y}}(i, j) = \int_{q_i}^{q_i + \Delta_x} \int_{r_j}^{r_j + \Delta_y} \mathbf{p}_{x,y}(xy) dx dy \approx \Delta_x \Delta_y \mathbf{p}_{x,y}(q_i, r_j),$$

and the kernel size is made small so that the kernel functions approach delta functions (although the squared kernel functions do not do so). In other words, the limit in the kernel size is taken *before* the limit in the grid size. We can then write population expression for the kernel generalised variance, in the limit of small kernel size, as

$$\begin{aligned} &\lim_{\sigma_x, \sigma_y \rightarrow 0} I(\mathbf{k}_l; \mathbf{l}_l) \\ &= \lim_{\sigma_x, \sigma_y \rightarrow 0} -\frac{1}{2} \log \left(\left| \mathbf{I} - \left(\mathbf{E}_{x,y}(\mathbf{k}_l \mathbf{l}_l^\top) - \mathbf{E}_x(\mathbf{k}_l) \mathbf{E}_y(\mathbf{l}_l^\top) \right)^\top \left(\mathbf{E}_x(\mathbf{k}_l \mathbf{k}_l^\top) - \mathbf{E}_x(\mathbf{k}_l) \mathbf{E}_x(\mathbf{k}_l^\top) \right)^{-1} \right. \right. \\ &\quad \left. \left. \times \left(\mathbf{E}_{x,y}(\mathbf{k}_l \mathbf{l}_l^\top) - \mathbf{E}_x(\mathbf{k}_l) \mathbf{E}_y(\mathbf{l}_l^\top) \right) \left(\mathbf{E}_y(\mathbf{l}_l \mathbf{l}_l^\top) - \mathbf{E}_y(\mathbf{l}_l) \mathbf{E}_y(\mathbf{l}_l^\top) \right)^{-1} \right| \right) \\ &\approx \lim_{\sigma_x, \sigma_y \rightarrow 0} -\frac{1}{2} \log \left(\left| \mathbf{I} - \left(\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top \right)^\top \left(\frac{\Delta_x}{2\sigma_x\sqrt{\pi}} \mathbf{D}_x - \mathbf{p}_x \mathbf{p}_x^\top \right)^{-1} \right. \right. \\ &\quad \left. \left. \times \left(\mathbf{P}_{xy} - \mathbf{p}_x \mathbf{p}_y^\top \right) \left(\frac{\Delta_y}{2\sigma_y\sqrt{\pi}} \mathbf{D}_y - \mathbf{p}_y \mathbf{p}_y^\top \right)^{-1} \right| \right) \\ &= 0, \end{aligned}$$

where we use the expression for the squared kernel in (57). In other words, $I(\mathbf{k}_l; \mathbf{l}_l)$ does *not* approach $I(\hat{x}; \hat{y})$ as the kernel size decreases. This problem reveals the need to enforce the opposite assumption to that made above, namely $\sigma_x \gg \Delta_x$ and $\sigma_y \gg \Delta_y$ (see Section 3.1.4).³⁹

We conclude this section with a brief discussion of the link between the empirical estimate of $I(\mathbf{k}_l; \mathbf{l}_l)$ and the KGV. As described by Bach and Jordan (2002a) and by Gretton (2003, Section 9.2.3, Appendix D.5.2), an empirical estimate of $I(\mathbf{k}_l; \mathbf{l}_l)$ is obtained via the usual expression (9), where ρ_i are the solutions to the generalised eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{K}_l \mathbf{H}(\mathbf{L}_l)^\top \\ \mathbf{L}_l \mathbf{H}(\mathbf{K}_l)^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \rho_i \begin{bmatrix} \mathbf{K}_l \mathbf{H}(\mathbf{K}_l)^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_l \mathbf{H}(\mathbf{L}_l)^\top \end{bmatrix} \begin{bmatrix} \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix}, \quad (58)$$

and \mathbf{K}_l and \mathbf{L}_l are defined in Section 3.1.4 (replacing the Parzen windows with the appropriate RKHS kernels). This is simply the kernel CCA problem, but with the solutions expressed in terms of linear combinations of the grid points \mathbf{q} and \mathbf{r} mapped into \mathcal{F} and \mathcal{G} , respectively. As the grid becomes infinitely fine, and assuming $k(\cdot, \cdot)$ and $l(\cdot, \cdot)$ to be continuous, we recover the standard kernel CCA formulation.⁴⁰

39. Also bear in mind that the expression for the KGV used in practice is defined in the limit of infinitely small grid size, but with finite kernel size, rather than vice versa. That said, the ratios $\frac{\Delta_x}{\sigma_x}$ and $\frac{\Delta_y}{\sigma_y}$ suggest a possible resolution of this convergence problem might be to decrease the kernel size and the grid spacing at the same time, as the number of samples rises.

40. This is not a proof - we would need to formally establish both convergence of the kernel CCA solutions in the limit of an infinitely fine grid size, and to demonstrate that the converged solutions lie in the span of the mapped data. These details fall outside the scope of the present study.

References

- S. Achard, D.-T. Pham, and C. Jutten. Blind source separation in post-nonlinear mixtures. In *3rd International Conference on ICA and BSS*, 2001.
- S. Achard, D.-T. Pham, and C. Jutten. Quadratic dependence measure for nonlinear blind source separation. In *4th International Conference on ICA and BSS*, 2003.
- S. Akaho. A kernel method for canonical correlation analysis. In *Proceedings of the International Meeting of the Psychometric Society (IMPS2001)*, 2001.
- S.-I. Amari, A. Cichoki, and Y. H. A new learning algorithm for blind signal separation. In *Advances in Neural Information Processing Systems*, volume 8, pages 757–763. MIT Press, 1996.
- F. Bach and M. Jordan. Kernel independent component analysis - (matlab code, version 1.1). <http://www.cs.berkeley.edu/~fbach/kernel-ica/index.htm>
- F. Bach and M. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002a.
- F. Bach and M. Jordan. Tree-dependent component analysis. In *Uncertainty in Artificial Intelligence*, volume 18, 2002b.
- C. R. Baker. Mutual information for gaussian processes. *SIAM Journal on Applied Mathematics*, 19(2):451–458, 1970.
- C. R. Baker. Joint measures and cross-covariance operators. *Transactions of the American Mathematical Society*, 186:273–289, 1973.
- G. Bakır, A. Gretton, M. Franz, and B. Schölkopf. Multivariate regression with stiefel constraints. Technical Report 101, Max Planck Institute for Biological Cybernetics, 2004.
- A. Bell and T. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.
- A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique using second order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–444, 1997.
- A. Belouchrani and M. G. Amin. Blind source separation based on time-frequency signal representations. *IEEE Transactions on Signal Processing*, 46(11):2888–2897, 1998.
- R. N. Bracewell. *The Fourier Transform and its Applications*. McGraw Hill, New York, 1986.
- L. Breiman and J. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80:580–598, 1985.
- J.-F. Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 90(8):2009–2026, 1998a.
- J.-F. Cardoso. Multidimensional independent component analysis. In *ICASSP*, 1998b.

- A. Chen and P. Bickel. Consistent independent component analysis and prewhitening. Technical report, Berkeley, 2004.
- A. Cichocki and S.-I. Amari. *Adaptive Blind Signal and Image Processing*. John Wiley and Sons, New York, 2002.
- P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36:287–314, 1994.
- T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, 1991.
- N. Cristianini, J. Shawe-Taylor, and J. Kandola. Spectral kernel methods for clustering. In *NIPS*, volume 14, Cambridge, MA, 2002. MIT Press.
- J. Dauxois and G. M. Nkiet. Nonlinear canonical analysis and independence tests. *Annals of Statistics*, 26(4):1254–1278, 1998.
- R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, second edition, 2001.
- A. Edelman, T. Arias, and S. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.
- S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.
- K. Fukumizu, F. Bach, and A. Gretton. Consistency of kernel canonical correlation analysis. Technical Report 942, Institute of Statistical Mathematics, 2005.
- K. Fukumizu, F. R. Bach, and M. I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.
- M. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic Press, London, 1984.
- A. Gretton. *Kernel Methods for Classification and Signal Separation*. PhD thesis, Cambridge University Engineering Department, 2003.
- A. Gretton, O. Bousquet, A. Smola, and B. Schoelkopf. Measuring statistical dependence with hilbert-schmidt norms. Technical Report 140, MPI for Biological Cybernetics, 2005a.
- A. Gretton, R. Herbrich, and A. Smola. The kernel mutual information. Technical report, Cambridge University Engineering Department and Max Planck Institute for Biological Cybernetics, 2003a.
- A. Gretton, R. Herbrich, and A. Smola. The kernel mutual information. In *ICASSP*, volume 4, pages 880–883, 2003b.
- A. Gretton, A. Smola, O. Bousquet, and R. Herbrich. Behaviour and convergence of the constrained covariance. Technical Report 130, MPI for Biological Cybernetics, 2004.

- A. Gretton, A. Smola, O. Bousquet, R. Herbrich, A. Belitski, M. Augath, Y. Murayama, J. Pauls, B. Schölkopf, and N. Logothetis. Kernel constrained covariance for dependence measurement. In *AISTATS*, volume 10, 2005b.
- D. Hardoon, J. Shawe-Taylor, and O. Friman. KCCA for fMRI analysis. In *Proceedings of Medical Image Understanding and Analysis*, London, 2004.
- S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15(5):1089–1124, 2003.
- S. Haykin. *Neural Networks : A Comprehensive Foundation*. Macmillan, New York, 2nd edition, 1998.
- M. Hein and O. Bousquet. Kernels, associated structures, and generalizations. Technical Report 127, Max Planck Institute for Biological Cybernetics, 2004.
- R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.
- S. Hosseni and C. Jutten. On the separability of nonlinear mixtures of temporally correlated sources. *IEEE Signal Processing Letters*, 10(2):43–46, 2003.
- A. Hyvärinen. One-unit contrast functions for independent component analysis: A statistical analysis. In *Proc. IEEE Neural Networks for Signal Processing Workshop*, pages 388–397, 1997.
- A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, New York, 2001.
- A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.
- A. Hyvärinen and M. Plumbley. Optimization with orthogonality constraints: a modified gradient method. Unpublished note, 2002.
- Y. I. Ingster. Asymptotically minimax testing of the hypothesis of independence. *Zap. Nauchn. Seminar. LOMI, 153 (1986) pp. 60-72, Translation in J. Soviet. Math.*, 44:466–476, 1989.
- J. Jacod and P. Protter. *Probability Essentials*. Springer, New York, 2000.
- M. Kuss. Kernel multivariate analysis. Master’s thesis, Technical University of Berlin, 2001.
- P. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(5):365–377, 2000.
- E. Learned-Miller and J. Fisher III. ICA using spacings estimates of entropy. *JMLR*, 4:1271–1295, 2003.
- T.-W. Lee, M. Girolami, A. Bell, and T. Sejnowski. A unifying framework for independent component analysis. *Computers and Mathematics with Applications*, 39:1–21, 2000.
- S. E. Leurgans, R. A. Moyeed, and B. W. Silverman. Canonical correlation analysis when the data are curves. *Journal of the Royal Statistical Society, Series B (Methodological)*, 55(3):725–740, 1993.

- T. Melzer, M. Reiter, and H. Bischof. Kernel canonical correlation analysis. Technical Report PRIP-TR-65, Pattern Recognition and Image Processing Group, TU Wien, 2001.
- E. Mourier. Éléments aléatoires dans un espace de Banach. *Ann. Inst. H. Poincaré Sect B.*, 161: 161–244, 1953.
- L. Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15:1191–1253, 2003.
- A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, New York, 1991.
- B. Pearlmutter. Music samples to illustrate the context-sensitive generalisation of ICA. <http://www.cs.unm.edu/~bap/demos.html>
- D.-T. Pham. Fast algorithms for mutual information based independent component analysis. *IEEE Transactions on Signal Processing*, 2002. Submitted.
- D.-T. Pham and P. Garat. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Transactions on Signal Processing*, 45(7):1712–1725, 1997.
- A. Rényi. On measures of dependence. *Acta Math. Acad. Sci. Hungar.*, 10:441–451, 1959.
- R. Rosipal and L. Trejo. Kernel partial least squares regression in reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 1(2):97–123, 2001.
- A. Samarov and A. Tsybakov. Nonparametric independent component analysis. *Bernoulli*, 10: 565–582, 2004.
- B. Schölkopf and A. Smola. *Learning with Kernels*. MIT press, Cambridge, MA, 2002.
- B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
- J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.
- B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, New York, 1986.
- I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *JMLR*, 2, 2001.
- H. Stögbauer, A. Kraskov, S. A. Astakhov, and P. Grassberger. Least dependent component analysis based on mutual information. *Phys. Rev. E*, 70(6):066123, 2004.
- A. Taleb and C. Jutten. Source separation in post-nonlinear mixtures. *IEEE Transactions on Signal Processing*, 47(10):2807–2820, 1999.
- F. Theis. Blind signal separation into groups of dependent signals using joint block diagonalisation. In *ISCAS*, pages 5878–5881, 2005.

- T. van Gestel, J. Suykens, J. de Brabanter, B. de Moor, and J. Vanderwalle. Kernel canonical correlation analysis and least squares support vector machines. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*. Springer Verlag, 2001.
- X.-L. Zhu and X.-D. Zhang. Adaptive RLS algorithm for blind source separation using a natural gradient. *IEEE Signal Processing Letters*, 9(12):432–435, 2002.
- L. Zwald, O. Bousquet, and G. Blanchard. Statistical properties of kernel principal component analysis. In *Proceedings of the 17th Conference on Computational Learning Theory (COLT)*, 2004.

Efficient Margin Maximizing with Boosting*

Gunnar Rätsch

*Friedrich Miescher Laboratory of the Max Planck Society
Spemannstrasse 35
72076 Tübingen, Germany*

GUNNAR.RAETSCH@TUEBINGEN.MPG.DE

Manfred K. Warmuth

*University of California at Santa Cruz
Santa Cruz, CA 95060, USA*

MANFRED@CSE.UCSC.EDU

Editor: John Shawe-Taylor

Abstract

AdaBoost produces a linear combination of base hypotheses and predicts with the sign of this linear combination. The linear combination may be viewed as a hyperplane in feature space where the base hypotheses form the features. It has been observed that the generalization error of the algorithm continues to improve even after all examples are on the correct side of the current hyperplane. The improvement is attributed to the experimental observation that the distances (margins) of the examples to the separating hyperplane are increasing even after all examples are on the correct side.

We introduce a new version of AdaBoost, called AdaBoost*, that explicitly maximizes the minimum margin of the examples up to a given precision. The algorithm incorporates a current estimate of the achievable margin into its calculation of the linear coefficients of the base hypotheses. The bound on the number of iterations needed by the new algorithms is the same as the number needed by a known version of AdaBoost that must have an explicit estimate of the achievable margin as a parameter. We also illustrate experimentally that our algorithm requires considerably fewer iterations than other algorithms that aim to maximize the margin.

1. Introduction

Boosting algorithms are greedy methods for forming linear combinations of base hypotheses. In the most common scenario the algorithm is given a fixed set of labeled training examples and in each iteration updates a distribution on these examples (i.e. a set of non-negative weights that sum to one). It then is given a *base* hypothesis whose weighted error (probability of wrong classification) is slightly below 50%. This base hypothesis is used to update the distribution on the examples: The algorithm increases the weights of those examples that were wrongly classified by the base hypothesis. At the end of each stage the base hypothesis is added to the linear combination, and the sign of this linear combination forms the current hypothesis of the boosting algorithm.

*, Part of this work was done while G. Rätsch was at Fraunhofer FIRST Berlin, at UC Santa Cruz, the Australian National University and the Max Planck Institute for biological Cybernetics. G. Rätsch was partially funded by DFG under contract JA 379/91, JA 379/71, MU 987/1-1 and by EU in the NeuroColt II project. M.K. Warmuth and visits of G. Rätsch to UC Santa Cruz were partially funded by the NSF grant CCR-9821087. G. Rätsch thanks S. Mika, S. Sonnenburg, S. Lemm and K.-R. Müller for discussions. M.K. Warmuth thanks J. Liao and Karen Glocer for their help.

The most well known boosting algorithm is AdaBoost (Freund and Schapire, 1997). It is "adaptive" in that the linear coefficient of the base hypothesis depends on the weighted error of the base hypothesis at the time when the base hypothesis was added to the linear combination. AdaBoost has two interesting properties. First, along with earlier boosting algorithms (Schapire, 1992; Freund, 1995), its training error has the following exponential convergence property: if the weighted training error of the t -th base hypothesis is $\epsilon_t = \frac{1}{2} - \frac{1}{2}\gamma_t$, then an upper bound on the training error of the signed linear combination is reduced by a factor of $1 - \frac{1}{2}\gamma_t^2$ at stage t . Second, it has been observed experimentally that AdaBoost continues to "learn" even after the training error of the signed linear combination is zero (Schapire et al., 1998). That is, in experiments the generalization error continues to improve. The signed linear combination can be viewed as a homogeneous *hyperplane* in a feature space, where each base hypothesis represents one feature or dimension. We define the *margin* of an example as a signed distance to the hyperplane times its \pm label (See Section 2 and Appendix A for precise definitions). As soon as the training error is zero, the examples are on the right side and all have positive margin. It has also been observed that the margins of the examples continue to increase even after the training error is zero. There are theoretical bounds on the generalization error of linear classifiers (e.g. Schapire et al., 1998; Breiman, 1999; Koltchinskii et al., 2001) that improve with the margin of the classifier, which is defined as the size of the minimum margin of the examples. Thus the fact that the margins improve experimentally seems to explain why AdaBoost still learns after the training error is zero.

There is one flaw in this argument: AdaBoost has not been proven to maximize the margin of the final hypothesis. We demonstrate this experimentally in Section 5. Moreover, Rudin et al. (2004a, 2005) recently showed that there are cases where AdaBoost provably does not maximize the margin. Breiman (1999) proposed a modified algorithm – called Arc-GV (**Arcing-Game Value**) – suitable for this task and showed that it *asymptotically* maximizes the margin. Similar results are shown in Grove and Schuurmans (1998) and Bennett et al. (2000). In this paper we present an algorithm that produces a final hypothesis with margin at least $\rho^* - v$, where ρ^* is the unknown maximum margin achievable by any convex combination of base hypotheses and v a precision parameter.

If we know ρ^* , then a linear combination with margin at least $\rho^* - v$ can be found by a parameterized version of AdaBoost called AdaBoost $_{\rho}$ (cf. Rätsch et al. (2001); Rätsch and Warmuth (2002)): When the parameter ρ of AdaBoost $_{\rho}$ is set to $\rho^* - v$, then after $\frac{2\ln N}{v^2}$ iterations, where N is the number of examples, the margin of the produced linear combination is guaranteed to be at least $\rho^* - v$. The case when ρ^* is not known is more difficult. In a preliminary conference paper (Rätsch and Warmuth, 2002) we used AdaBoost $_{\rho}$ iteratively in a binary search like fashion: $\log_2(2/v)$ calls to AdaBoost $_{\rho}$ are guaranteed to produce a margin at least $\rho^* - v$. All but the last call to AdaBoost $_{\rho}$ are used to find a suitable value of the parameter ρ and in the last call this parameter is used to create the final linear combination in at most $\frac{2\ln N}{v^2}$ iterations.

In this paper we greatly simplify our answer for the case when ρ^* is unknown. We have a new *one pass* algorithm called AdaBoost *_v that produces a linear combination with margin at least $\rho^* - v$ after $\frac{2\ln N}{v^2}$ iterations. Note that this is the same guarantee we had on the number of iterations of AdaBoost $_{\rho}$ when it used the theoretically optimal parameter $\rho = \rho^* - v$. The new algorithm AdaBoost *_v uses the parameter v and a *current estimate* of the achievable margin in the computation of the linear coefficients of the base learners.

Except for the algorithm presented in the previous conference paper, this is the first result on the fast convergence of a boosting algorithm to the maximum margin solution that works for all $\rho^* \in [-1, 1]$. Using previous results one can only show that AdaBoost *asymptotically* converges to

a final hypothesis with margin at least $\rho^*/2$ if $\rho^* > 0$ and if subtle conditions on the chosen base hypotheses are satisfied (cf. Corollary 5).

Recently other versions of AdaBoost have been published that are guaranteed to produce a linear combination of margin at least $\rho^* - \nu$ after $\Omega(\nu^{-3})$ iterations (Rudin et al., 2004c,b). Even though these algorithms have weaker iteration bounds than AdaBoost * , they were reported to perform better experimentally (Rudin et al., 2004c,a). We briefly compare AdaBoost $^*_\nu$ to these more recent algorithms and show that the better empirical performance was due to the wrong choice of ν .

The original AdaBoost was designed to find a final hypothesis of margin at least zero. Our algorithm maximizes the margin for all values of ρ^* . This includes the inseparable case (i.e. $\rho^* < 0$), where one minimizes the overlap between the two classes. In this case AdaBoost runs forever without necessarily increasing the margin. Our algorithm is also useful when the base hypotheses given to the Boosting algorithm are *strong* in the sense that they already separate the data and have margin greater than zero, but less than one. In this case $0 < \rho^* < 1$ and AdaBoost aborts immediately because the linear coefficients of such hypotheses become unbounded. In contrast, our new algorithm also maximizes the margin when presented with strong learners.

The big advantage of this algorithm is an absolute bound on the number of iterations: After $\frac{2\ln N}{\nu^2}$ iterations AdaBoost $^*_\nu$ is guaranteed to produce a hypothesis with margin at least $\rho^* - \nu$. Our algorithm is applicable in sophisticated settings where the number of hypotheses may be infinite. In Appendix B we use AdaBoost $^*_\nu$ to learn a convex combination of support vector kernels and show that the same guarantees hold on the number of iterations of the algorithm.

The paper is structured as follows: Section 2 introduces some basic notation and in Section 3 we first describe *AdaBoost $_\rho$* which requires a lower bound ρ of the maximum margin ρ^* as a parameter. Then we present our new algorithm *AdaBoost $^*_\nu$* , which is similar to *AdaBoost $_\rho$* , but continuously adapts ρ based on a precision parameter ν . Up to this point we stay at a high level of presentation with the goal of making our algorithms accessible to the quick reader. In Section 4 we introduce more notation and give a detailed analysis of both algorithms. First, we prove that if the weighted training error of the t -th base hypothesis is $\varepsilon_t = \frac{1}{2} - \frac{1}{2}\gamma_t$, then an upper bound on the fraction of examples with margin smaller than ρ is reduced by a factor of $1 - \frac{1}{2}(\rho - \gamma_t)^2$ at stage t of *AdaBoost $_\rho$* (cf. Section 4.2) (A slightly improved factor is shown for the case when $\rho > 0$). However, to achieve a large margin one needs to assume that the guess ρ is smaller than ρ^* . For the latter case we prove an exponential convergence rate of *AdaBoost $_\rho$* . Then we discuss a method for automatically tuning ρ depending on the errors of the base hypotheses and a precision parameter ν . We show that after roughly $\frac{2\ln N}{\nu^2}$ iterations our new one-pass algorithm *AdaBoost $^*_\nu$* is guaranteed to produce a linear combination with margin at least $\rho^* - \nu$. This strengthens the results of our preliminary conference paper (Rätsch and Warmuth, 2002), which had an additional $\log_2(2/\nu)$ factor in the total number times the weak learner is called and much higher constants. In Section 5, we compare the algorithms experimentally and discuss heuristics for tuning ν in Section 5.2. Finally we briefly summarize and discuss our results in the Conclusion Section.

2. Preliminaries and Basic Notation

We consider the standard two-class supervised machine learning problem: Given a set of N i.i.d. training examples (\mathbf{x}_n, y_n) , $n = 1, \dots, N$, with $\mathbf{x}_n \in \mathcal{X}$ and $y_n \in \mathcal{Y} := \{-1, +1\}$, we would like to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that is able to generalize well on unseen data generated from the same distribution as the training data.

In the case of ensemble learning (like boosting), there is a fixed underlying set of *base* hypotheses $\mathcal{H} := \{h \mid h : \mathcal{X} \rightarrow [-1, 1]\}$ from which the ensemble is built. For now we only assume that \mathcal{H} is finite, but we will show in Section 4.5 that this assumption can be dropped in most cases and that all of the following analysis also applies to the case of infinite hypothesis sets.

Boosting algorithms iteratively form non-negative linear combinations of hypotheses from \mathcal{H} . In each iteration t , a base hypothesis $h_t \in \mathcal{H}$ with a non-negative coefficient α_t is added to the linear combination. We denote the combined hypothesis as follows (Note that we normalized the weights):

$$\tilde{f}_\alpha(\mathbf{x}) = \text{sign } f_\alpha(\mathbf{x}), \text{ where } f_\alpha(\mathbf{x}) = \sum_{t=1}^T \frac{\alpha_t}{\sum_{r=1}^T \alpha_r} h_t(\mathbf{x}), h_t(\mathbf{x}) \in \mathcal{H}, \text{ and } \alpha_t \geq 0 .$$

The “black box” that selects the base hypothesis in each iteration is called the *weak* learner. For AdaBoost, it has been shown that if the weak learner is guaranteed to select base hypotheses of weighted error slightly below 50%, then the combined hypothesis is consistent with the training set in a small number of iterations (Freund and Schapire, 1997). We will discuss bounds on the number of iterations in detail in Section 4. Since at most one new base hypothesis is added in each iteration, the size of the final hypothesis is bounded by the number of iterations. These bounds are important because the sample size bounds provable in the PAC model grow with the size of the final hypothesis (Schapire, 1992; Freund, 1995).

In more recent research (Schapire et al., 1998) it was also shown that a bound on the generalization error decreases with the size of the margin of the final hypothesis f . The margin of a single example (\mathbf{x}_n, y_n) w.r.t. f is defined as $y_n f_\alpha(\mathbf{x}_n)$. Thus the margin quantifies by how far this example is on the y_n side of the hyperplane \tilde{f} . In Appendix A we clarify how the margin of an example is related to its ℓ_∞ -distance to the hyperplane with normal α . The margin of the combined hypothesis f is the *minimum margin* of all N examples. The goal of this paper is to find a small non-negative linear combination of base hypotheses from \mathcal{H} with margin close to the maximum achievable margin.

The following table gives some of the main notations that will be used throughout this paper:

Symbol	Description
n, N	index and number of examples
m, M	index and number of hypotheses if finite
t, T	index and number of iterations
\mathcal{X}	input space
\mathcal{Y}	label space $\{\pm 1\}$
(\mathbf{x}, y)	an example and its label
\mathcal{H}, h_m	set of base hypotheses and the m -th element
α	hypothesis weight vector
\mathbf{d}	weighting on the training set
$\mathbf{I}(\cdot)$	the indicator function: $\mathbf{I}(true) = 1$ and $\mathbf{I}(false) = 0$
ρ	the margin parameter of AdaBoost $_\rho$
$\{\rho_t\}$	the sequence of margin parameters of AdaBoost $_{\{\rho_t\}}$
ρ^*	the maximum margin
$\hat{\rho}_t$	margin in the t -th iteration
v	the accuracy parameter of AdaBoost $_v^*$
ε	weighted classification error
γ^*	the minimum edge

Symbol	Description
γ	an arbitrary edge threshold
γ_t	the edge of the t -th hypothesis

3. AdaBoost $_{\rho}$ and AdaBoost $_{\gamma}^*$

The original AdaBoost was designed to find a consistent hypothesis \tilde{f} which is defined as a signed linear combination f with margin greater zero. We start with a slight modification of AdaBoost, which finds (if possible) a linear combination of base learners with margin ρ , where ρ is a parameter (cf. Algorithm 1).¹ We call this algorithm AdaBoost $_{\rho}$, as it naturally generalizes AdaBoost for the case when the *target margin* is ρ . The original AdaBoost algorithm now becomes AdaBoost $_0$.

Algorithm 1: – The AdaBoost $_{\rho}$ algorithm – with margin parameter ρ

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \rangle$, No. of iterations T , margin target ρ
 2. **Initialize:** $d_n^1 = \frac{1}{N}$ for all $n = 1 \dots N$
 3. **Do for** $t = 1, \dots, T$,
 - (a) Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto [-1, 1]$
 - (b) Calculate the edge γ_t of h_t : $\gamma_t = \sum_{n=1}^N d_n^t y_n h_t(\mathbf{x}_n)$
 - (c) **if** $|\gamma_t| = 1$, **then** $\alpha_1 = \text{sign}(\gamma_t)$, $h_1 = h_t$, $T = 1$; **break**
 - (d) Set $\alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t} - \frac{1}{2} \ln \frac{1 + \rho}{1 - \rho}$
 - (e) Update weights: $d_n^{t+1} = \frac{d_n^t \exp(-\alpha_t y_n h_t(\mathbf{x}_n))}{Z_t}$,
where $Z_t = \sum_{n=1}^N d_n^t \exp(-\alpha_t y_n h_t(\mathbf{x}_n))$
 4. **Output:** $f_{\alpha}(\mathbf{x}) = \sum_{t=1}^T \frac{\alpha_t}{\sum_{r=1}^T \alpha_r} h_t(\mathbf{x})$
-

The algorithm AdaBoost $_{\rho}$ was already known as *unnormalized Arcing* (Breiman, 1999) or *AdaBoost-type Algorithm* (Rätsch et al., 2001). Moreover, it is related to algorithms proposed in Freund and Schapire (1999) and Zhang (2002). The only difference from AdaBoost is the choice of the hypothesis coefficients α_t : An additional term $-\frac{1}{2} \ln \frac{1+\rho}{1-\rho}$ appears in the expression for the hypothesis coefficient α_t . This term vanishes when $\rho = 0$. The parameter ρ can be seen as a *guess* of the maximum margin ρ^* . If ρ is chosen properly (slightly below ρ^*), then AdaBoost $_{\rho}$ will converge exponentially fast to a combined hypothesis with nearly the maximum margin. See Section 4.2 for details.

1. The original AdaBoost algorithm was formulated in terms of weighted training error ϵ_t of a base hypothesis. Here we use an equivalent more convenient formulation in terms of the edge γ_t , where $\epsilon_t = \frac{1}{2} - \frac{1}{2}\gamma_t$ (cf. Section 4.1).

The following example illustrates how AdaBoost_ρ works. Assume the weak learner returns the constant hypothesis $h_t(\mathbf{x}) \equiv 1$. The weighted error of this hypothesis is the sum of all negative weights, i.e. $\varepsilon_t = \sum_{y_n=-1} d_n^t$ and its edge is $\gamma_t = 1 - 2\varepsilon_t$. The coefficient α_t is chosen so that the edge of h_t with respect to the new distribution is exactly ρ (instead of 0 as for the original AdaBoost). More precisely, the given choice of α_t assures that this edge is ρ only for ± 1 -valued base hypotheses.

For a more general base hypothesis h_t with continuous range $[-1, +1]$, choosing α_t such that Z_t as a function of α_t is minimized, guarantees that the edge of h_t with respect to the distribution \mathbf{d}^{t+1} is ρ . See Schapire and Singer (1999) for a similar discussion. Choosing α_t as in step 3 (d) approximately minimizes Z_t when the range of h_t is $[-1, +1]$.

In Kivinen and Warmuth (1999) and Lafferty (1999), the standard boosting algorithms are interpreted as approximate solutions to the following optimization problem: choose a distribution \mathbf{d} of maximum entropy subject to the constraints that the edges of the previous hypotheses are *equal* to zero. In this paper we use the *inequality* constraints that the edges of the previous hypotheses are at most ρ . The α_t 's function as Lagrange multipliers for these inequality constraints. Since $g(x) = \frac{1}{2} \ln \frac{1+x}{1-x}$ is an increasing function,

$$\alpha_t = \frac{1}{2} \ln \frac{1+\gamma_t}{1-\gamma_t} - \frac{1}{2} \ln \frac{1+\rho}{1-\rho} \geq 0 \quad \text{iff} \quad \gamma_t \geq \rho . \tag{1}$$

Notice that when $\rho = 0$, adding h_t or $-h_t$ leads to the same distribution \mathbf{d}^{t+1} . This symmetry is broken for $\rho \neq 0$.

Since one does not know the value of the optimum margin ρ^* is not known beforehand, one also needs to find ρ^* . In Rätsch and Warmuth (2002) we presented the *Marginal AdaBoost* algorithm which constructs a sequence $\{\rho_r\}_{r=1}^R$ converging to ρ^* . A fast way to find a real value up to a certain accuracy v in the interval $[-1, 1]$ is a *binary search* since one needs only $\log_2(2/v)$ steps.² Thus the previous Marginal AdaBoost algorithm uses AdaBoost_{ρ_r} (Algorithm 1) to decide whether the current guess ρ_r is *larger* or *smaller* than ρ^* . Depending on the outcome, ρ_r can be chosen so that the region of uncertainty for ρ^* is roughly cut in half. However, in the previous algorithm all but the last of the $\log_2(2/v)$

In this paper we propose a different algorithm, called AdaBoost_v^{*}. Here $v > 0$ is a precision parameter. The algorithm finds a non-negative linear combination with margin at least $\rho^* - v$. Like Arc-GV (Breiman, 1999), the new algorithm essentially runs AdaBoost_ρ once but instead of using a fixed margin estimate ρ , it updates ρ in an appropriate way. We shall show iteration bounds for our algorithm AdaBoost_v^{*} which are not known for Arc-GV. The latter algorithm produces an essentially³ monotonically increasing sequence of margin estimates, while in AdaBoost_v^{*} we use a monotonically decreasing sequence. The improved sequence of estimates is based on two new theoretical insights, which will be developed in the next section.

We will show that the number of iterations required by the new one-pass AdaBoost_v^{*} algorithm (see Algorithm 2 for pseudo-code) is at most $\frac{2 \ln N}{v^2}$. This equals the iteration bound for the best algorithm we know of for the case when ρ^* is known and we seek a linear combination of margin at least $\rho^* - v$: AdaBoost_ρ with parameter $\rho = \rho^* - v$. The iteration bound for the new algorithm is the same as the iteration bound for the last call to AdaBoost_ρ of the previous Marginal AdaBoost algorithm.

2. If one knows that $\rho^* \in [a, b]$, one needs only $\log_2((b-a)/v)$ steps.
 3. In the original formulation the sequence was not necessarily increasing, but Rätsch (2001) showed that it leads to the same result and easier proofs if one restricts it to be monotonically increasing.

Algorithm 2: – The AdaBoost_v^{*} algorithm – with accuracy parameter v

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \rangle$, No. of Iterations T , desired accuracy v
 2. **Initialize:** $d_n^1 = 1/N$ for all $n = 1 \dots N$
 3. **Do for** $t = 1, \dots, T$,
 - (a) Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto [-1, 1]$
 - (b) Calculate the edge γ_t of h_t : $\gamma_t = \sum_{n=1}^N d_n^t y_n h_t(\mathbf{x}_n)$
 - (c) **if** $|\gamma_t| = 1$, **then** $\alpha_t = \text{sign}(\gamma_t)$, $h_1 = h_t$, $T = 1$; **break**
 - (d) $\gamma_t^{\min} = \min_{r=1, \dots, t} \gamma_r$; $\rho_t = \gamma_t^{\min} - v$
 - (e) Set $\alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t} - \frac{1}{2} \ln \frac{1 + \rho_t}{1 - \rho_t}$
 - (f) Update weights: $d_n^{t+1} = \frac{d_n^t \exp(-\alpha_t y_n h_t(\mathbf{x}_n))}{Z_t}$,
where $Z_t = \sum_{n=1}^N d_n^t \exp(-\alpha_t y_n h_t(\mathbf{x}_n))$
 4. **Output:** $f_\alpha(\mathbf{x}) = \sum_{t=1}^T \frac{\alpha_t}{\sum_{r=1}^T \alpha_r} h_t(\mathbf{x})$
-

4. Detailed Analysis

In this section we are going to analyze the algorithms in detail. We start by showing the relationship between optimal edges and margins, prove and illustrate the convergence properties of AdaBoost_p and finally prove the convergence of AdaBoost_v^{*}.

4.1 Weak learning and margins

The standard assumption made on the weak learning algorithm for the PAC analysis of Boosting algorithm is that the weak learner returns a hypothesis h from a fixed set \mathcal{H} that is slightly better than random guessing. That is, that the error rate ε is consistently smaller than $\frac{1}{2}$. Note that the error rate of $\frac{1}{2}$ could easily be reached by a fair coin, assuming both classes have the same prior probabilities. More formally, the error ε of a ± 1 valued hypothesis is defined as the fraction of examples that are misclassified. In Boosting this is extended to weighted example sets and the error is defined as

$$\varepsilon_h(\mathbf{d}) = \sum_{n=1}^N d_n \mathbf{I}(y_n \neq h(\mathbf{x}_n)),$$

where h is the hypothesis returned by the weak learner and \mathbf{I} is the indicator function with $\mathbf{I}(\text{true}) = 1$ and $\mathbf{I}(\text{false}) = 0$. The distribution $\mathbf{d} = (d_1, \dots, d_N)$ of the examples is such that $d_n \geq 0$ and $\sum_{n=1}^N d_n = 1$.

When the range of a hypothesis h is the entire interval $[-1, +1]$, then the *edge* $\gamma_h(\mathbf{d}) = \sum_{n=1}^N d_n y_n h(\mathbf{x}_n)$ is a more convenient quantity for measuring the quality of h . This edge is an affine transformation of the error for the case when h has range ± 1 : $\varepsilon_h(\mathbf{d}) = \frac{1}{2} - \frac{1}{2}\gamma_h(\mathbf{d})$ and $\varepsilon_h(\mathbf{d}) \leq \frac{1}{2}$ iff $\gamma_h(\mathbf{d}) \geq 0$.

Recall from Section 2 that the margin of a given example (\mathbf{x}_n, y_n) is defined as $y_n f_\alpha(\mathbf{x}_n)$. Also recall that \mathcal{H} is the set from which the weak learner chooses its base hypotheses. Assume for a moment that \mathcal{H} is finite. If we combine all hypotheses from \mathcal{H} , then the following well-known theorem establishes the connection between margins and edges (first seen in connection with Boosting in Freund and Schapire, 1996; Breiman, 1999):⁴

Theorem 1 (Min-Max-Theorem, von Neumann (1928))

$$\gamma^* := \min_{\mathbf{d}} \max_{m=1, \dots, M} \sum_{n=1}^N d_n y_n h_m(\mathbf{x}_n) = \max_{\alpha} \min_{n=1, \dots, N} y_n \sum_{m=1}^M \alpha_m h_m(\mathbf{x}_n) =: \rho^*, \quad (2)$$

where $\mathbf{d} \in \mathcal{P}^N$, $\alpha \in \mathcal{P}^M$ and $M = |\mathcal{H}|$. Here \mathcal{P}^k denotes the k -dimensional probability simplex.

Thus, the minimum edge γ^* that can be achieved over all possible distributions \mathbf{d} of the training set is equal to the maximum margin ρ^* of any linear combination of hypotheses from \mathcal{H} . Also, for any non-optimal distributions \mathbf{d} and hypothesis weights α we always have

$$\max_{h \in \mathcal{H}} \gamma_h(\mathbf{d}) > \gamma^* = \rho^* > \min_{n=1, \dots, N} y_n f_\alpha(\mathbf{x}_n).$$

In particular, if the weak learning algorithm is guaranteed to return a hypothesis with edge at least γ for any distribution on the examples, then $\gamma^* \geq \gamma$ and by the above duality there exists a combined hypothesis with margin at least γ . If γ is equal to its upper bound γ^* then there exists a combined hypothesis with margin exactly $\gamma = \rho^*$ that only uses hypotheses that are actually returned by the weak learner in response to certain distributions on the examples.

From this discussion we can derive a sufficient condition on the weak learning algorithm to reach the maximum margin (for the case when \mathcal{H} finite). If the weak learner returns hypotheses whose edges are at least γ^* , then there exists a linear combination of these hypotheses that attains a margin $\gamma^* = \rho^*$. We will prove later that our AdaBoost_v algorithm efficiently finds a linear combination with margin close to ρ^* (cf. Theorem 6).

Constraining the edges of the previous hypotheses to equal zero (as done in the *totally corrective algorithm* of Kivinen and Warmuth (1999)) leads to a problem if there is no solution satisfying these constraints. At the end of trial t , the set of previous hypotheses is $\mathcal{H}_t = \{h_1, \dots, h_t\}$ and the totally corrective algorithm finds a distribution such that $\gamma_h(\mathbf{d}) = 0$, for all $h \in \mathcal{H}_t$. Because of the above duality and the fact that $\mathcal{H}_t \subseteq \mathcal{H}$,

$$\gamma_t^* := \min_{\mathbf{d}} \max_{h \in \mathcal{H}_t} \gamma_h(\mathbf{d}) \leq \gamma^* = \rho^* .$$

The non-decreasing sequence (γ_t^*) converges to ρ^* from below. If $\rho^* > 0$, then the equality constraints on the edges are not satisfiable as soon as $\gamma_t^* > 0$.

In contrast our new algorithm AdaBoost_v is motivated by a system of inequality constraints $\gamma_h(\mathbf{d}) \leq \rho$, for $h \in \mathcal{H}_t$, where ρ is adapted. Again, if $\rho < \rho^*$, then the system of inequalities with this

4. This is a zero-sum game with payoff matrix $y_n h_m(\mathbf{x}_n)$. The row player finds a mixture \mathbf{d} over the rows/examples and the column player a mixture α over the column/hypotheses. Adding a row/example makes the minimax value of the game go down and adding a column/hypothesis makes it go up.

$\hat{\rho}$ may not have a solution (and the Lagrange multipliers may diverge to infinity). In AdaBoost $_{\rho}^*$ we start with ρ large and decrease it when necessary. As we shall see, the algorithm maintains a margin parameter ρ that is always at least $\rho^* - v$.

4.2 Convergence properties of AdaBoost $_{\rho}$

Let AdaBoost $_{\{\rho_t\}}$ denote the version of AdaBoost $_{\rho}$ that uses a time varying margin parameter ρ_t at iteration t . Thus in step 3 (d) of the algorithm, ρ is replaced by ρ_t . This extension will be necessary for the later analysis of AdaBoost $_{\rho}^*$. The sequences $\{\rho_t\}_{t=1}^T$ might be specified while running the algorithm. For instance, in the algorithm Arc-GV, Breiman (1999) chooses ρ_t as $\min_{n=1, \dots, N} y_n f_{\alpha_{t-1}}(\mathbf{x}_n)$. Breiman (1999) showed that Arc-GV *asymptotically* converges to the maximum margin (see discussion in next section). In the following we answer the question how to best choose the sequence $\{\rho_t\}$ so as to optimize bounds on the fraction of examples which have a margin at most ρ .

Lemma 2 *For any $\rho \in [-1, 1]$, the final hypothesis f_{α} of AdaBoost $_{\{\rho_t\}}$ satisfies the following inequality:*

$$\frac{1}{N} \sum_{n=1}^N \mathbf{I}(y_n f_{\alpha}(\mathbf{x}_n) \leq \rho) \leq \left(\prod_{t=1}^T Z_t \right) \exp \left\{ \sum_{t=1}^T \rho \alpha_t \right\} = \prod_{t=1}^T \exp \{ \rho \alpha_t + \ln Z_t \} \quad (3)$$

where $Z_t = \sum_{n=1}^N d_n^t \exp(-\alpha_t y_n h_t(\mathbf{x}_n))$ and $\alpha_t = \frac{1}{2} \ln \frac{1+\gamma_t}{1-\gamma_t} - \frac{1}{2} \ln \frac{1+\rho_t}{1-\rho_t}$.

The proof directly follows from a simple extension of Theorem 1 in Schapire and Singer (1999) (see also Schapire et al. (1998)).

We now use a lemma from Rätsch et al. (2001) to upper bound the right hand side (rhs) of the above inequality:

Lemma 3 *Let γ_t be the edge of h_t in the t -th iteration of AdaBoost $_{\{\rho_t\}}$. Assume $-1 \leq \rho_t \leq \gamma_t$. Then for all $\rho \in [-1, 1]$,*

$$\exp \{ \rho \alpha_t + \ln Z_t \} \leq \exp \left(-\frac{1+\rho}{2} \ln \left(\frac{1+\rho_t}{1+\gamma_t} \right) - \frac{1-\rho}{2} \ln \left(\frac{1-\rho_t}{1-\gamma_t} \right) \right). \quad (4)$$

Note that this generalizes Theorem 5 of (Freund and Schapire, 1997) to the case when the target margin is not zero.

AdaBoost $_{\{\rho_t\}}$ makes progress, if the rhs of (4) is smaller than one. Suppose we would like to reach a margin ρ on all training examples, where we obviously need to assume $\rho \leq \rho^*$. We can then ask which sequence of $\{\rho_t\}_{t=1}^T$ one should use to find such combined hypothesis in as few iterations as possible. The rhs of (4) can be rewritten as

$$\exp(\Delta_2(\rho, \rho_t) - \Delta_2(\rho, \gamma_t)),$$

where $\Delta_2(a, b) := \frac{1+a}{2} \ln \frac{1+a}{1+b} + \frac{1-a}{2} \ln \frac{1-a}{1-b}$ denotes the binary relative entropy between $a, b \in [-1, 1]$. Therefore the rhs of (4) is minimized for $\rho_t = \rho$ (independent of γ_t) and one should always use this constant choice.

This means that when $\rho_t = \rho$ then the rhs of (4) is reduced by a factor of $\exp(-\Delta_2(\rho, \gamma_t))$, which can be upper bounded by inspecting the Taylor expansion at $\gamma_t = \rho$ and noticing that when $0 \leq \rho < \gamma_t$, all terms of order three and higher are negative:

$$\exp(-\Delta_2(\rho, \gamma_t)) < 1 - \frac{1}{2} \frac{(\rho - \gamma_t)^2}{1 - \rho^2}, \text{ for } 0 \leq \rho < \gamma_t. \quad (5)$$

The denominator $1 - \rho^2$ speeds up the convergence when $\rho \gg 0$. Notice that when $\rho = 0$, then we recover the original AdaBoost bound.

Now we determine an upper bound on the number of iterations needed by AdaBoost $_{\rho}$ for achieving a margin of ρ on all examples, given that the maximum margin is ρ^* :

Corollary 4 *Assume the weak learner always returns a base hypothesis with an edge $\gamma_t \geq \rho^*$. If $0 \leq \rho \leq \rho^* - \nu$, $\nu > 0$, then AdaBoost $_{\rho}$ will converge to a solution with margin at least ρ on all examples in at most $\frac{2 \ln N (1 - \rho^2)}{\nu^2}$ iterations.*

Proof By Lemma 2 and (4), (5):

$$\frac{1}{N} \sum_{n=1}^N \mathbf{I}(y_n f(\mathbf{x}_n) \leq \rho) < \prod_{t=1}^T \left(1 - \frac{1}{2} \frac{(\rho - \gamma_t)^2}{1 - \rho^2} \right) \leq \left(1 - \frac{1}{2} \frac{\nu^2}{1 - \rho^2} \right)^T.$$

The margin is at least ρ for all examples, if the rhs is smaller than $\frac{1}{N}$; hence after at most

$$\frac{\ln N}{-\ln \left(1 - \frac{1}{2} \frac{\nu^2}{1 - \rho^2} \right)} \leq \frac{2 \ln N (1 - \rho^2)}{\nu^2}$$

iterations, which proves the statement. ■

When $\rho < 0$, then inequality (5) can be replaced with the following weaker inequality which holds for all distinct $\rho, \gamma_t \in [-1, 1]$:

$$\exp(-\Delta_2(\rho, \gamma_t)) < \exp\left(-\frac{1}{2}(\rho - \gamma_t)^2\right). \tag{6}$$

This leads to the same bound as in the above corollary except that the factor $(1 - \rho^2)$ is omitted. Thus when $\rho < 0$, the bound on the number of iterations becomes $\frac{2 \ln N}{\nu^2}$ (Rätsch, 2001, page 25).

4.3 Asymptotic Margin of AdaBoost $_{\rho}$

With the methods shown so far we can determine the asymptotic value of margin of the hypothesis produced by the original AdaBoost algorithm. First, we state a lower bound on the margin that is achieved by AdaBoost $_{\rho}$. There is a gap between this lower bound and the upper bound of Theorem 1. In a second part we consider an experiment that shows that depending on some subtle properties of the weak learner, the margin of combined hypotheses generated by AdaBoost can converge to quite different values (while the maximum margin is kept constant). We observe that the previously lower bound on the margin is quite tight in empirical cases.

As long as each factor in the rhs of Eq. (3) is smaller than 1, the bound decreases. If the factor is at most $1 - \mu$ and $\mu > 0$, then the rhs converges exponentially fast to zero. The following corollary considers the asymptotic case and gives a lower bound on the margin.

Corollary 5 (Rätsch (2001)) *Assume AdaBoost $_{\rho}$ generates hypothesis h_1, h_2, \dots with edges $\gamma_1, \gamma_2, \dots$ and coefficients $\alpha_1, \alpha_2, \dots$. Let $\gamma^{\min} = \inf_{t=1,2,\dots} \gamma_t$ and assume $\gamma^{\min} > \rho$. Furthermore, let*

$$\hat{\rho}_t = \min_{n=1,\dots,N} \frac{y_n \sum_{r=1}^t \alpha_r h_r(\mathbf{x}_n)}{\sum_{r=1}^t \alpha_r}$$

be the achieved margin in the t -th iteration and $\hat{\rho} = \sup_{t=1,2,\dots} \hat{\rho}_t$. Then the margin $\hat{\rho}$ of the combined hypothesis is bounded from below by

$$\hat{\rho} \geq \frac{\ln(1 - \rho^2) - \ln(1 - (\gamma^{\min})^2)}{\ln\left(\frac{1 + \gamma^{\min}}{1 - \gamma^{\min}}\right) - \ln\left(\frac{1 + \rho}{1 - \rho}\right)}. \quad (7)$$

From (7) one can understand the interaction between ρ and γ^{\min} : If the difference between γ^{\min} and ρ is small, then the rhs of (7) is small. Thus, if ρ with $\rho \leq \gamma^{\min}$ is large, then $\hat{\rho}$ must be large, i.e. choosing a larger ρ results in a larger margin on the training examples. A Taylor expansion of the rhs of (7) shows that the margin is lower bounded by $\frac{\gamma^{\min} + \rho}{2}$. This known lower bound (Breiman, 1999, Theorem 7.2) is greater than ρ if $\gamma^{\min} > \rho$.

In Section 4.1 we reasoned that $\gamma^{\min} \leq \rho^*$. If the parameter AdaBoost_ρ is chosen too small, then we guarantee only that the margin of the produced linear combination converges asymptotically to a value at below ρ^* . In the original formulation of AdaBoost we have $\rho = 0$ and we guarantee only that AdaBoost_0 achieves a margin of at least $\frac{\gamma^{\min} + \rho}{2} = \frac{1}{2}\gamma^{\min}$. This shortfall in the margin provable for AdaBoost motivates our new AdaBoost_v^* which is guaranteed to optimize the margin.

4.3.1 EXPERIMENTAL ILLUSTRATION OF COROLLARY 5

To illustrate the above-mentioned gap, we perform an experiment showing how tight (7) can be. We analyze two different settings: (i) the weak learner selects the hypothesis with largest edge over all hypotheses (i.e. the best case) and (ii) the weak learner selects the hypothesis with minimum edge among all hypotheses with edge larger than ρ^* (i.e. the worst case). Corollary 5 holds for both cases since the weak learner is allowed to return *any* hypothesis with edge larger than ρ^* .

We use random data with N training examples, where N is drawn uniformly between 10 and 200. The labels are drawn at random from a binomial distribution with equal probability. We use a hypothesis set with 10^4 random hypotheses with range $\{+1, -1\}$. We first choose a parameter p uniformly in $(0,1)$. Then the label of each hypothesis on each example is chosen to agree with the label of the example with probability p .⁵ First we compute the solution ρ^* of the margin-LP problem via the left hand side of (2). Then we compute the combined hypothesis generated by AdaBoost_ρ after 10^4 iterations for $\rho = 0$ and $\rho = \frac{1}{3}$ using the best and the worst selection strategy, respectively. The latter algorithm depends on ρ^* . We chose 300 hypothesis sets based on 300 random draws of p . The random choice of p ensures that there are cases with small and large optimal margins. For each hypothesis set we did two runs of AdaBoost_ρ using the best and worst selection strategies. The result of each run is represented as a point in Figure 1. The abscissa is the maximum achievable margin ρ^* for each run. The ordinate is the margin of AdaBoost_ρ using the best (green) and the worst strategy (red).

There is a large difference between these selection strategies. Whereas the margin of the worst strategy is *tightly* lower bounded by (7), the best strategy has near maximum margin. These experiments show that one obtains different results by changing the selection strategy of the weak learning algorithm. Our lower bound holds for both selection strategies. The looseness of the bounds is indeed a problem, as we cannot predict where AdaBoost_ρ converges to.⁶ However, note that moving $\hat{\rho}$ closer to ρ^* reduces the gap (see also Figure 1 [right]).

5. We do not allow duplicate hypotheses or hypotheses that agree with the labels on all examples.

6. One might even be able to construct cases where the outputs are not at all converging.

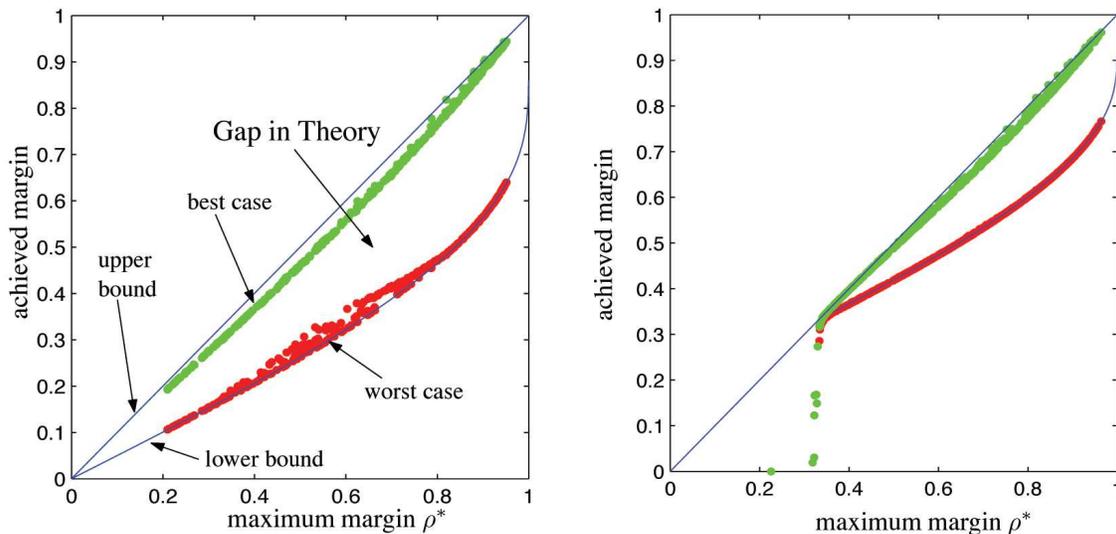


Figure 1: Achieved margins of AdaBoost_ρ using the best (green) and the worst (red) selection on random data for $\rho = 0$ [left] and $\rho = \frac{1}{3}$ [right]. On the abscissa is the maximum achievable margin ρ^* and on the ordinate the margin achieved by AdaBoost_ρ for one data realization. For comparison we plotted the upper bound $y = x$ and the lower bound (7). On the interval $[\rho, 1]$, there is a clear gap between the performance of the worst and best selection strategies. The margin of the worst strategy is very close to the lower bound (7) and the best strategy has near maximum margin. If ρ is chosen slightly below the maximum achievable margin then this gap is reduced to 0.

Recently, it has been shown by Rudin et al. (2005) that there exist cases where the weighting \mathbf{d}^t on the examples cycles indefinitely between non-optimal solutions. This proves that AdaBoost does not generally maximize the margin. Furthermore, it was shown in Rudin et al. (2004b) that the gap exhibited in Figure 1 is not an experimental artifact: under certain conditions the lower bound (7) was proven to be tight.

4.3.2 DECREASING THE STEP SIZE

Breiman (1999) conjectured that the inability to maximize the margin is due to the fact that the normalized hypothesis coefficients may “circulate endlessly through the convex set”, which is defined by the lower bound on the margin. In fact, motivated from our previous experiments, it seems possible to implement a weak learner that appropriately switches between optimal and worst case performance, leading to non-convergent normalized hypothesis coefficients.

Rosset et al. (2002) have shown that AdaBoost with infinitesimally small step sizes may maximize the margin, if the weak learner uses the best selection strategy. This is similar to what we found empirically for finite step sizes and motivates us to analyze AdaBoost_ρ with step sizes chosen as follows:

$$\hat{\alpha}_t = \eta \alpha_t = \frac{\eta}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t} - \frac{\eta}{2} \ln \frac{1 + \rho}{1 - \rho},$$

for some $\eta > 0$. For $\eta = 1$ we recover AdaBoost $_{\rho}$. Following the same proof technique as for Corollary 5, we can show that under the same conditions as given in Corollary 5

$$\hat{\rho} \geq \frac{-\ln((1+\gamma)\exp(-\hat{\alpha}) + (1-\gamma)\exp(\hat{\alpha}))}{\hat{\alpha}},$$

where $\hat{\alpha} = \frac{\eta}{2} \ln \frac{1+\gamma}{1-\gamma} - \frac{\eta}{2} \ln \frac{1+\rho}{1-\rho}$. Note that if η goes to zero, then $\hat{\rho} = \gamma$. Interestingly, this is independent of the choice of ρ . Thus if the weak learner always returns hypotheses with edges $\gamma_t \geq \rho^*$ ($t = 1, 2, \dots$), where ρ^* is the maximum margin, then by the Min-Max Theorem, the margin is maximized when η goes to zero. However, there are no guarantees on the convergence speed.

4.4 Convergence of AdaBoost $_{\nu}^*$

The AdaBoost $_{\nu}^*$ algorithm is based on two insights:

- According to the discussion after Lemma 3, the most rapid convergence to a combined hypothesis with margin $\rho^* - \nu$ occurs for AdaBoost $_{\rho}$ when one chooses ρ_t as close as possible to $\rho^* - \nu$.
- For distributions on the examples that are hard for the weak learner (i.e. the weak learner achieves a small edge), the edge γ_t will be close to ρ^* .

The idea is that by choosing $\rho_t = (\min_{r=1, \dots, t} \gamma_r) - \nu$ we concentrate on the hardest distribution we generated so far and can so find a *close* overestimate of $\rho^* - \nu$. This forces an acceleration of the convergence to a large margin and leads to distributions for which the weak learner has to return small edges.

Note that if the weak learner always returns hypotheses with edge $\gamma_t = \rho^*$ which is the worst case under the assumption that $\gamma_t \geq \rho^*$, then $\rho_t = \rho^* - \nu$ in each iteration. In this case the same smallest step size is taken in every iteration which is determined by ρ^* and ν . This smallest step size decreases with the desired accuracy ν , which matches the intuition from Section 4.3.2 that decreasing the step size achieves larger and therefore more accurate margins.

We will now state and prove our main theorem:

Theorem 6 *Assume the weak learner always returns a base hypothesis with an edge $\gamma_t \geq \rho^*$. Then after $\frac{2 \ln N}{\nu^2}$ iterations AdaBoost $_{\nu}^*$ (Algorithm 2) is guaranteed to produce a combined hypothesis f of margin at least $\rho^* - \nu$.*

Proof Let $\rho = \rho^* - \nu$ be the margin that we would like to achieve. By assumption on the performance of the weak learner, $\rho^* \leq \min_{r=1, \dots, T} \gamma_r = \gamma_T^{\min}$ and thus $\rho = \rho^* - \nu \leq \gamma_T^{\min} - \nu$. In step 3 (d) of Algorithm 2, ρ_t was set to $\gamma_t^{\min} - \nu$. Hence $\rho \leq \rho_t$ for each iteration.

Lemmas 2 and 3 imply that

$$\frac{1}{N} \sum_{n=1}^N \mathbf{I}(y_n f(\mathbf{x}_n) \leq \rho) \leq \prod_{t=1}^T \exp \left(-\frac{1+\rho}{2} \ln \left(\frac{1+\rho_t}{1+\gamma_t} \right) - \frac{1-\rho}{2} \ln \left(\frac{1-\rho_t}{1-\gamma_t} \right) \right)$$

We now rewrite the rhs using $\alpha_t = \frac{1}{2} \ln \frac{1+\gamma_t}{1-\gamma_t} - \frac{1}{2} \ln \frac{1+\rho_t}{1-\rho_t}$:

$$= \prod_{t=1}^T \exp \left(-\frac{1}{2} \ln \left(\frac{1+\rho_t}{1+\gamma_t} \right) - \frac{1}{2} \ln \left(\frac{1-\rho_t}{1-\gamma_t} \right) + \rho \alpha_t \right)$$

By (1), $\alpha_t \geq 0$ since $\rho_t \leq \gamma_t$. By replacing ρ by its upper bound ρ_t we get:

$$\leq \prod_{t=1}^T \exp\left(-\frac{1+\rho_t}{2} \ln\left(\frac{1+\rho_t}{1+\gamma_t}\right) - \frac{1-\rho_t}{2} \ln\left(\frac{1-\rho_t}{1-\gamma_t}\right)\right)$$

Finally, by (6) we have:

$$= \prod_{t=1}^T \exp(-\Delta(\rho_t, \gamma_t)) < \prod_{t=1}^T \exp\left(-\frac{(\rho_t - \gamma_t)^2}{2}\right) \leq \exp\left(-\frac{Tv^2}{2}\right).$$

is at most $\frac{1}{N}$, then by the above chain of inequalities, $\frac{1}{N} \sum_{n=1}^N \mathbf{I}(y_n f(\mathbf{x}_n) \leq \rho) < \frac{1}{N}$ and the margin of each of the N examples is at least ρ . The theorem now follows from the fact that $\frac{1}{N} < \exp\left(-\frac{1}{2}Tv^2\right)$, if the number of iterations T is at least $\frac{2\ln N}{v^2}$. ■

If one assumes $\rho_t \geq 0$, then Theorem 6 could be improved by a factor of $(1 - \rho_t^2)$ in each iteration, using the refined upper bound of Corollary 4. Since $\rho_t \geq \rho^* - v$, one would obtain the bound $\frac{\ln N(1 - (\rho^* - v)^2)}{v^2}$ if $\rho^* \geq v$, but this factor will only matter for very large margins.

4.5 Infinite Hypothesis Sets

So far we have implicitly assumed that the hypothesis space is finite. In this section we will show that this assumption is (often) not necessary. Also note, if the output of the hypotheses is discrete, the hypothesis space is effectively finite (Rätsch et al., 2002). For *infinite hypothesis sets*, Theorem 1 can be restated in a weaker form as:

Theorem 7 (Weak Min-Max, e.g. Nash and Sofer (1996))

$$\gamma^* := \min_{\mathbf{d}} \sup_{h \in \mathcal{H}} \sum_{n=1}^N y_n h(\mathbf{x}_n) d_n \geq \sup_{\alpha} \min_{n=1, \dots, N} y_n \sum_{q: \alpha_q > 0} \alpha_q h_q(\mathbf{x}_n) =: \rho^*, \quad (8)$$

where $\mathbf{d} \in \mathcal{P}^N$, $\alpha \in \mathcal{P}^{|\mathcal{H}|}$ with finite support.

We call $\Gamma = \gamma^* - \rho^*$ the “duality gap”. In particular for any $\mathbf{d} \in \mathcal{P}^N$, $\sup_{h \in \mathcal{H}} \sum_{n=1}^N y_n h(\mathbf{x}_n) d_n \geq \gamma^*$ and for any $\alpha \in \mathcal{P}^{|\mathcal{H}|}$ with finite support, $\min_{n=1, \dots, N} y_n \sum_{q: \alpha_q \geq 0} \alpha_q h_q(\mathbf{x}_n) \leq \rho^*$.

In theory the duality gap may be nonzero. However, Lemma 3 and Theorem 6 do not assume finite hypothesis sets and show that the margin will converge arbitrarily close to ρ^* , as long as the weak learning algorithm can return a hypothesis in each iteration that has an edge not smaller than ρ^* .

In other words, the duality gap may result from the fact that the sup on the left side cannot be replaced by a max, i.e. there might not exist a *single* hypothesis h with edge larger or equal to ρ^* . By assuming that the weak learner is always able to pick good enough hypotheses ($\geq \rho^*$), one automatically gets by Lemma 3 that $\Gamma = 0$.

Under certain conditions on \mathcal{H} this maximum always exists and strong duality holds (for details see e.g. Rätsch et al., 2002; Rätsch, 2001; Hettich and Kortanek, 1993; Nash and Sofer, 1996):

Theorem 8 (Strong Min-Max) *If the set of vectors $\{(h(\mathbf{x}_1), \dots, h(\mathbf{x}_N)) \mid h \in \mathcal{H}\}$ is compact, then $\Gamma = 0$.*

In general, this requirement can be fulfilled by the weak learning algorithms whose outputs continuously depend on the distribution \mathbf{d} . Furthermore, the outputs of the hypotheses need to be bounded (cf. step 3a in AdaBoost $_{\rho}$). The first requirement might be a problem with weak learning algorithms that are some variants of decision stumps or decision trees. However, there is a simple trick to avoid this problem: Roughly speaking, at each point with discontinuity $\hat{\mathbf{d}}$, one adds all hypotheses to \mathcal{H} that are limit points of $L(S, \mathbf{d}^s)$, where $\{\mathbf{d}^s\}_{s=1}^{\infty}$ is an arbitrary sequence converging to $\hat{\mathbf{d}}$ and $L(S, \mathbf{d})$ denotes the hypothesis returned by the weak learning algorithm for distribution \mathbf{d} and training sample S (Rätsch, 2001). This procedure assures that \mathcal{H} is closed.

The above theorem is applied in Appendix B to obtain iteration bounds for AdaBoost $_{\nu}^*$ in the context of learning a convex combination of support vector kernels.

5. Experimental Comparison

In this section we discuss two experiments: The first one shows that our theoretical bounds can be tight on artificial data and the second one compares our algorithm to the one proposed in Rudin et al. (2004a).

5.1 Illustration on Toy Examples

We are aware that maximizing the margin of the ensemble does not lead to improved generalization performance in all cases. In fact for fairly noisy data sets the opposite has been reported (cf. Quinlan, 1996; Breiman, 1999; Grove and Schuurmans, 1998; Rätsch et al., 2001). Also, Breiman (1998) reported an example where the margins of all examples are larger in one ensemble than another and the latter generalized considerably better.

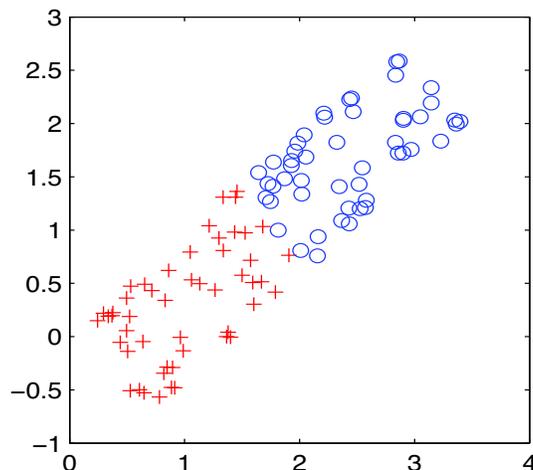


Figure 2: The two *discriminative dimensions* of our separable one hundred dimensional data set.

Nonetheless, the theoretical bounds on the generalization error of linear classifiers improves with the margin. We therefore expect to be able to measure differences in the generalization error between a function that maximizes the margin and one that does not. Similar results have been obtained in Schapire et al. (1998) on a multi-class optical character recognition problem.

Here we report experiments on artificial data to illustrate how our algorithm works and how it compares to AdaBoost. Our data is 100 dimensional and contains 98 nuisance dimensions with uniform noise. The other two dimensions are plotted exemplary in Figure 2. For training we use only 100 examples which means that controlling the capacity of the ensemble is essential.

As the weak learning algorithm we use C4.5 decision trees provided by Quinlan (1992) using an option to control the number of nodes in the tree. We have tuned C4.5 to generate trees with about three nodes. Otherwise, the weak learner often classifies all training examples correctly and over-fits the data already. Furthermore, since in this case the margin is already maximum (equal to 1), boosting algorithms would stop since $\gamma = 1$. We therefore need to limit the complexity of the weak learner, in good agreement with the bounds on the generalization error (Schapire et al., 1998).

Moreover, we have to deal with the fact that C4.5 cannot use weighted samples. We therefore use weighted bootstrapping (e.g. Efron and Tibshirani, 1994). However, this amplifies the problem that the resulting hypotheses might in some cases have an edge smaller than the maximum margin, which according to the Min-Max-Theorem should not occur if the weak learner performs optimally. We deal with this problem by repeatedly calling C4.5 on different bootstrap realizations if the edge is smaller than the margin of the current linear combination. Furthermore, for AdaBoost_v^{*}, a small edge of one hypothesis can spoil the margin estimate ρ_t . We address this problem by resetting $\rho_t = \hat{\rho}_t + v$, whenever $\rho_t \leq \hat{\rho}_t$, where $\hat{\rho}_t$ is the margin of the currently combined hypothesis.

In Figure 3 we see a typical run of AdaBoost, Marginal AdaBoost, AdaBoost_v^{*} and Arc-GV for $v = .1$. For comparison we plot the margins of the hypotheses generated by AdaBoost (cf. Figure 3 (left)). One observes that it is not able to achieve a large margin efficiently. After 1000 iterations $\hat{\rho} = .37$.

Marginal AdaBoost as proposed in Rätsch and Warmuth (2002) proceeds in stages and first tries to find an estimate of the margin using a binary search. It calls AdaBoost _{ρ} three times. The first call of AdaBoost _{ρ} for $\rho = 0$ stops after four iterations because it has generated a consistent combined hypothesis. The lower bound l on ρ^* as computed by Marginal AdaBoost is $l = .07$ and the upper bound u is $.94$. The second time ρ is chosen to be in the middle of the interval $[l, u]$ and AdaBoost _{ρ} reaches the margin of $\rho = .51$ after 80 iterations. The interval is now $[.51, .77]$. Because the length of the interval $u - l = .27$ is small enough, Marginal AdaBoost leaves the loop through an exit condition, calls AdaBoost _{ρ} the last time for $\rho = u - v = .41$ and finally achieves the margin of $.55$.

In a run of Arc-GV for thousand iterations we observe a margin of the combined hypothesis of $.53$, while for our new algorithm, AdaBoost_v^{*}, we find $.58$. In this case the margin for AdaBoost_v^{*} is larger than the margins of all other algorithms when executed for one thousand iterations. It starts with slightly lower margins in the beginning, but then catches up due the better choice of the margin estimate.

	C4.5	AdaBoost	Marginal AdaBoost	AdaBoost _v [*]
E_{gen}	$7.4 \pm .11\%$	$4.0 \pm .11\%$	$3.6 \pm .10\%$	$3.5 \pm .10\%$
$\hat{\rho}$	—	$.31 \pm .01$	$.55 \pm .01$	$.58 \pm .01$

Table 2: Estimated generalization performances and margins with confidence intervals for decision trees (C4.5), AdaBoost, Marginal AdaBoost and AdaBoost_v^{*} on the toy data. All numbers are averaged over 200 splits into 100 training and 19900 test examples.

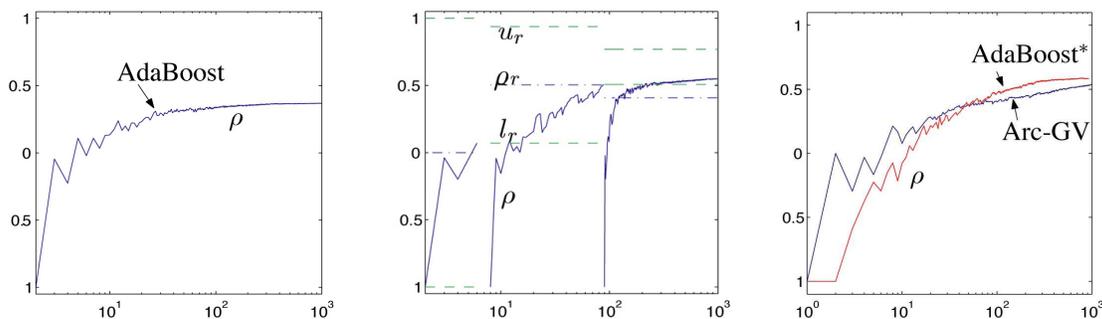


Figure 3: Illustration of the achieved margin of AdaBoost_0 (left), Marginal AdaBoost_v (middle), Arc-GV, and AdaBoost_v^* (right) at each iteration. Marginal AdaBoost_v calls AdaBoost_ρ three times while adapting ρ (dash-dotted). We also plot the values for l and u as in Marginal AdaBoost (dashed). (For details see Ratsch and Warmuth, 2002) AdaBoost_v^* achieves larger margins than AdaBoost . Compared to Arc-GV it starts slower, but then catches up in the later iterations. Here the correct choice of the parameter ρ is important.

In Table 2 we see the average performances of the four classifiers. For AdaBoost and AdaBoost_v^* we combined 200 hypotheses for the final prediction. For Marginal AdaBoost we use $v = .1$ and let the algorithm combine only 200 hypotheses for the final prediction to get a more fair comparison. We see a large improvement of all ensemble methods over the single classifier. There is also a slight, but – according to a t -test with confidence level 98% – significant difference between the generalization performances of AdaBoost and Marginal AdaBoost as well as AdaBoost and AdaBoost_v^* . Note also that the margins of the combined hypothesis achieved by Marginal AdaBoost and AdaBoost_v^* are on average almost twice as large as for AdaBoost . The difference in generalization performance between AdaBoost_v^* and Marginal AdaBoost is not statistically significant.

The differences between the achieved margins of both algorithms seem slightly significant (96%). The slightly larger margins generated by Marginal AdaBoost can be attributed to the fact that it uses many more calls to the weak learner than AdaBoost_v^* and after an estimate of the achievable margin is available, it starts optimizing the linear combination using this estimate.

It would be natural to use a two-pass algorithm: In the first pass use AdaBoost_v^* to get a margin estimate ρ size at least $\rho^* - v$ and then use this estimate in a final run of AdaBoost_ρ . The hypothesis produced in the second pass should have a larger margin and use fewer base hypotheses.

5.2 Heuristics for Tuning the Precision Parameter v

Our main results says that after $\frac{2\ln N}{v^2}$ iterations AdaBoost_v^* produces a hypothesis of margin at least $\rho^* - v$. Thus if the algorithm is allowed to run for T iterations, then v should be set to $v_T = \sqrt{\frac{2\ln N}{T}}$. If v is chosen much larger than v_T , then after T iterations AdaBoost_v^* often achieves a margin below $\rho^* - v_T$. Similarly, if v is chosen much smaller than v_T , then AdaBoost_v^* starts too slowly and after T iterations its margin is typically again below $\rho^* - v_T$.

Recently, Rudin et al. (2004a,c) proposed an algorithm, called *Coordinate Ascent Boosting*, which solves the same problem as AdaBoost_v^* . Their analysis of the algorithm shows that it needs

at most $\Omega(v^{-3})$ iterations to achieve a margin of at least $\rho^* - v$. While this theoretical result is clearly inferior to the guarantees which we provide for AdaBoost_v^* , their experimental evaluation of the algorithms seemed to suggest that the algorithm requires significantly fewer iterations than AdaBoost_v^* in practice. However, their observations were only due to the improper choice of the accuracy parameter v for AdaBoost_v^* : For $v = 10^{-3}$ (as chosen in their study), AdaBoost_v^* would need millions of iterations to achieve a guaranteed margin $\rho^* - v$. However, only the first 20K iterations were displayed and in this range their algorithms achieve a larger margin. For $T = 20K$ and $N = 50$, the precision parameter prescribed by our bounds is $v_T = .02$. When this parameter is used, then AdaBoost^* clearly beats all the other related algorithms (cf. Figure 4). We leave it to the reader to explore other heuristics for tuning v based on the theoretical results of this paper (See also the discussion at the end of the last subsection).

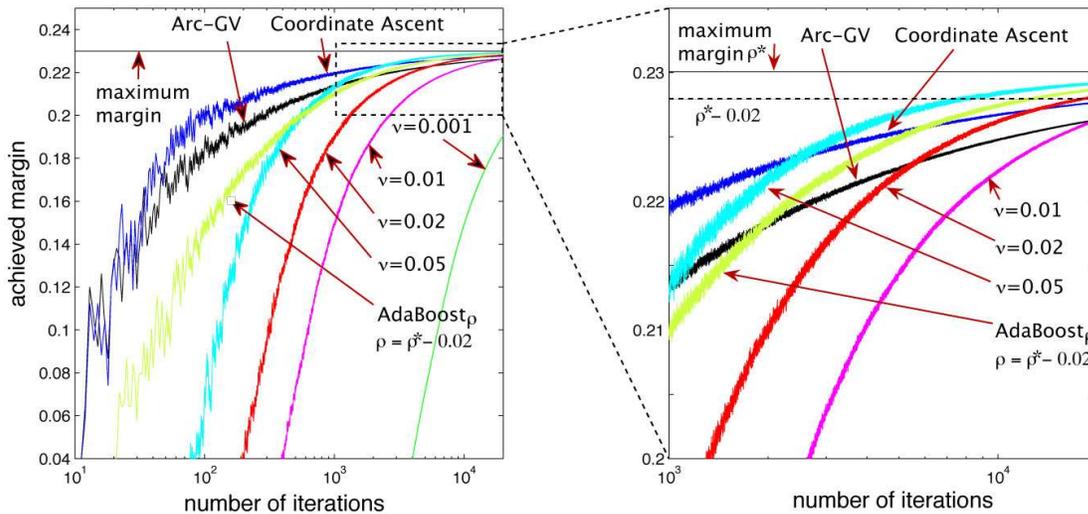


Figure 4: AdaBoost_v^* with different choices of v is compared to Arc-GV and the Coordinate Ascent Algorithm on the same artificial dataset 1 used in Rudin et al. (2004c) (We reconstructed this dataset from a figure given in Rudin et al. (2004b)): The number of iterations is $T = 20K$, the dimension of the examples is $N = 50$, and we assume that the base learner returns a hypothesis with maximum edge. If v is set to a reasonably close range around the value $v_T = .02$ prescribed by our bound, then AdaBoost_v^* achieves the margin which is significantly larger than the margins achieved by the other algorithms. If $v = .001 \ll v_T$ as chosen in Rudin et al. (2004c), then AdaBoost_v^* starts too slowly. In the case when the base learner returns a random hypothesis with edge only at least as large as ρ^* , then our algorithm compares even more favorably (not shown).

6. Conclusion

We have analyzed a generalized version of AdaBoost in the context of large margin algorithms. From von Neumann's Min-Max theorem we know that if the weak learner always returns a hypothesis with weighted classification error less than $\frac{1}{2} - \frac{1}{2}\gamma$ then the maximum achievable margin ρ^* is at least γ . The asymptotic analysis lead us to a lower bound on the margin of the final hypotheses generated by AdaBoost_p, which was shown to be rather tight in empirical cases. Our results indicate that vanilla AdaBoost generally does not maximize the margin, and only achieves a margin of about half the optimum.

To overcome these problems we provided an algorithm AdaBoost_v^{*} with the following provable guarantees: It produces a linear combination with margin at least $\rho^* - v$ and the number of base hypotheses used in this linear combination is at most $\frac{2 \ln n}{v^2}$. The new algorithm decreases its estimate ρ of the margin iteratively, such that the gap between the best and the worst case becomes arbitrarily small. Our analysis did not require additional properties of the weak learning algorithm. In simulation experiments we have illustrated the validity of our theoretical analysis.

Appendix A. Margins

First recall the definition of margin used in this paper, which is defined for a fixed set of examples $\{(\mathbf{x}_n, y_n) : 1 \leq n \leq N\}$ and a set of hypotheses $\mathcal{H} = \{h_1, \dots, h_M\}$ (here finite for the sake of simplicity):

$$\rho^*(\mathcal{H}) = \max_{\alpha} \min_{n=1, \dots, N} y_n \sum_{m=1}^M \alpha_m h_m(\mathbf{x}_n), \text{ where } \alpha \text{ is on the simplex } \mathcal{P}^M.$$

Note that we minimize over the margins of individual examples and maximize over the hyperplanes. Define the one-norm margin $\rho_1^*(\mathcal{H})$ in the same way but now α lies in the larger set $\{\alpha : \alpha \in \mathbb{R}^M \text{ and } \|\alpha\|_1 = 1\}$. It is well known that for a fixed example (\mathbf{x}_n, y_n) and normal $\alpha \in \mathbb{R}^M$, the one-norm margin $\frac{\sum_{m=1}^M \alpha_m h_m(\mathbf{x}_n)}{\sum_{m=1}^M |\alpha_m|}$ is the minimum ℓ_∞ -distance of the example to the hyperplane with normal α (Mangasarian, 1999; Rätsch et al., 2002), where the latter distance is defined as

$$\inf_{\mathbf{z} \in \mathbb{R}^M \text{ s.t. } \alpha \cdot \mathbf{z} = 0} y_n \max_{m=1, \dots, M} |h_m(\mathbf{x}_n) - z_m|.$$

Note that in this appendix, margins are defined as a function of the the hypotheses set \mathcal{H} because we will vary this set in a moment. Let $\text{cl}(\mathcal{H})$ be the closure of \mathcal{H} under negation, i.e. $\text{cl}(\mathcal{H}) = \mathcal{H} \cup \{-h : h \in \mathcal{H}\}$. Now, the following relationships are straightforward:

1. $\rho^*(\mathcal{H}) \leq \rho_1^*(\mathcal{H})$, $\rho^*(\text{cl}(\mathcal{H})) \geq 0$, and $\rho^*(\text{cl}(\mathcal{H})) \geq \rho_1^*(\mathcal{H})$.
2. If $\rho^*(\text{cl}(\mathcal{H})) > 0$, then $\rho^*(\text{cl}(\mathcal{H})) = \rho_1^*(\mathcal{H})$.
3. If $\rho_1^*(\mathcal{H}) \geq 0$, then $\rho^*(\text{cl}(\mathcal{H})) = \rho_1^*(\mathcal{H})$.

In summary, if the one-norm margin of \mathcal{H} is non-negative, then the margin of the closed hypotheses class $\text{cl}(\mathcal{H})$ coincides with the one-norm margin.

Appendix B. An Application to Multiple Kernel Learning

Sonnenburg et al. (2005) proposed a new algorithm for solving the multiple kernel learning (MKL) problem that was introduced in Lanckriet et al. (2004); Bach et al. (2004). The idea of MKL is to find a convex combination of J support vector kernels $k_j : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ ($j = 1, \dots, J$) that maximizes the SVM soft margin (cf. Bach et al. (2004)). In Sonnenburg et al. (2005) the original quadratically-constrained quadratic program was reformulated to the following semi-infinite linear program:

$$\min_{\beta \in \mathcal{P}^J} \sup_{\alpha \in \mathcal{A}} \sum_{j=1}^J \beta_j S_j(\alpha) \quad (9)$$

where

$$S_j(\alpha) := -\frac{1}{2} \sum_{r,s=1}^N \alpha_r \alpha_s y_r y_s k_j(\mathbf{x}_r, \mathbf{x}_s) + \sum_{n=1}^N \alpha_n$$

$$\mathcal{A} := \left\{ \alpha \mid \alpha \in \mathbb{R}^N, \mathbf{0} \leq \alpha \leq \mathbf{1}C, \sum_{n=1}^N y_n \alpha_n = 0 \right\}$$

and C is the SVM regularization constant. Note that this problem has infinitely many constraints: one for every vector α in its domain \mathcal{A} . Note that problem (9) is of the same type as the semi-infinite programming problem (8) which can be solved with AdaBoost_v^{*} (cf. discussion in Section 4.5). Since the $S_j(\alpha)$ are continuous functions and \mathcal{A} is compact, it follows from Theorem 8 that the duality gap is zero.

When AdaBoost_v^{*} is applied to this problem, a hypothesis with large edge has to be found in each iteration. In this case the hypotheses are α vectors and the edge is

$$\sum_{j=1}^J \beta_j S_j(\alpha) = -\frac{1}{2} \sum_{r,s} \alpha_r \alpha_s y_r y_s \left(\sum_{j=1}^J \beta_j k_j(\mathbf{x}_r, \mathbf{x}_s) \right) + \sum_i \alpha_i.$$

It has been noted that the edge in this case is nothing else than the negative SVM objective function for the combined kernel $k(\mathbf{x}_r, \mathbf{x}_s) = \sum_{j=1}^J \beta_j k_j(\mathbf{x}_r, \mathbf{x}_s)$. Hence, identifying an α vector with maximum edge amounts to solving the vanilla SVM quadratic optimization problem. Fortunately, many efficient SVM packages are available to solve this problem. Thus, the MKL problem can be efficiently solved using AdaBoost_v^{*} and our iteration bound for AdaBoost_v^{*} is applicable.

References

- F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Twenty-first international conference on Machine learning*. ACM Press, 2004. ISBN 1-58113-828-5.
- K. P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In P. Langley, editor, *Proceedings, 17th ICML*, pages 65–72, San Francisco, 2000. Morgan Kaufmann.
- L. Breiman. Are margins relevant in voting? Talk at the NIPS'98 workshop on Large Margin Classifiers, December 1998.

- L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1518, 1999. Also Technical Report 504, Statistics Department, University of California Berkeley, Dec. 1997.
- B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistic and Applied Probability*. Chapman and Hall/CRC, New York, 1994.
- Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2): 256–285, September 1995.
- Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.
- Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.
- A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.
- R. Hettich and K. O. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, September 1993.
- J. Kivinen and M. Warmuth. Boosting as entropy projection. In *Proc. 12th Annu. Conference on Comput. Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.
- V. Koltchinskii, D. Panchenko, and F. Lozano. Some new bounds on the generalization error of combined classifiers. In *Advances in Neural Information Processing Systems*, volume 13, 2001.
- J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 125–133, New York, NY, 1999. ACM Press.
- G. R. G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 2004.
- O. L. Mangasarian. Arbitrary-norm separating plane. *Operation Research Letters*, 24(1):15–23, 1999.
- S. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, NY, 1996.
- J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.
- J. R. Quinlan. Boosting first-order learning. *Lecture Notes in Computer Science*, 1160:143, 1996.
- G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany, October 2001.
- G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1-3):193–221, 2002. Special Issue on New Methods for Model Selection and Model Combination. Also NeuroCOLT2 Technical Report NC-TR-2000-085.

- G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller. Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE PAMI*, 24(9), September 2002. In press. Earlier version is GMD TechReport No. 119, 2000.
- G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3): 287–320, March 2001. Also NeuroCOLT Technical Report NC-TR-1998-021.
- G. Rätsch and M. K. Warmuth. Maximizing the margin with boosting. In *Proc. COLT*, volume 2375 of *LNAI*, pages 319–333, Sydney, 2002. Springer.
- S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin separator. Technical report, Department of Statistics, Stanford University, 2002.
- C. Rudin, I. Daubechies, and R. E. Schapire. On the dynamics of boosting. In *Advances in Neural Information Processing*, volume 15, 2004a.
- C. Rudin, I. Daubechies, and R. E. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 2005.
- C. Rudin, R. E. Schapire, and I. Daubechies. Analysis of boosting algorithms using the smooth margin function: A study of three algorithms. Unpublished manuscript, October 2004b.
- C. Rudin, R. E. Schapire, and I. Daubechies. Boosting based on a smooth margin. In *Proc. COLT'04*, LNCS. Springer Verlag, July 2004c.
- R. E. Schapire. *The Design and Analysis of Efficient Learning Algorithms*. PhD thesis, MIT Press, 1992.
- R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.
- R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.
- S. Sonnenburg, G. Rätsch, and C. Schäfer. Learning interpretable svms for biological sequence analysis. In *Proc. RECOMB'05*, LNCS. Springer Verlag, 2005.
- J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Math. Ann.*, 100:295–320, 1928.
- T. Zhang. Sequential greedy approximation for certain convex optimization problems. Technical report, IBM T. J. Watson Research Center, 2002.

On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning

Petros Drineas

*Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180, USA*

DRINEP@CS.RPI.EDU

Michael W. Mahoney

*Department of Mathematics
Yale University
New Haven, CT 06520, USA*

MAHONEY@CS.YALE.EDU

Editor: Nello Cristianini

Abstract

A problem for many kernel-based methods is that the amount of computation required to find the solution scales as $O(n^3)$, where n is the number of training examples. We develop and analyze an algorithm to compute an easily-interpretable low-rank approximation to an $n \times n$ Gram matrix G such that computations of interest may be performed more rapidly. The approximation is of the form $\tilde{G}_k = CW_k^+C^T$, where C is a matrix consisting of a small number c of columns of G and W_k is the best rank- k approximation to W , the matrix formed by the intersection between those c columns of G and the corresponding c rows of G . An important aspect of the algorithm is the probability distribution used to randomly sample the columns; we will use a judiciously-chosen and data-dependent nonuniform probability distribution. Let $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the spectral norm and the Frobenius norm, respectively, of a matrix, and let G_k be the best rank- k approximation to G . We prove that by choosing $O(k/\epsilon^4)$ columns

$$\|G - CW_k^+C^T\|_\xi \leq \|G - G_k\|_\xi + \epsilon \sum_{i=1}^n G_{ii}^2,$$

both in expectation and with high probability, for both $\xi = 2, F$, and for all $k : 0 \leq k \leq \text{rank}(W)$. This approximation can be computed using $O(n)$ additional space and time, after making two passes over the data from external storage. The relationships between this algorithm, other related matrix decompositions, and the Nyström method from integral equation theory are discussed.¹

Keywords: kernel methods, randomized algorithms, Gram matrix, Nyström method

1. Introduction

In this introductory section, we first, in Section 1.1, provide a summary of relevant background, then in Section 1.2 we summarize our main result, and finally, in Section 1.3, we provide an outline of the remainder of the paper.

1. A preliminary version of this paper appeared as Drineas and Mahoney (2005b,a).

1.1 Background

Given a collection \mathcal{X} of data points, which are often but not necessarily elements of \mathbb{R}^m , techniques such as linear support vector machines (SVMs), Gaussian processes (GPs), principal components analysis (PCA), and the related singular value decomposition (SVD), identify and extract structure from \mathcal{X} by computing linear functions, i.e., functions in the form of dot products, of the data. For example, in PCA the subspace spanned by the first k eigenvectors is used to give a k dimensional model of the data with minimal residual; thus, it provides a low-dimensional representation of the data. Such spectral analysis has a rich theoretical foundation and has numerous practical applications.

In many cases, however, there is nonlinear structure in the data (or the data, such as text, may not support the basic linear operations of addition and scalar multiplication). In these cases, kernel-based learning methods have proved to be quite useful (Cristianini and Shawe-Taylor, 2000; Schölkopf, Smola, and Müller, 1998). Kernel-based learning methods are a class of statistical learning algorithms, the best known examples of which are SVMs (Cristianini and Shawe-Taylor, 2000). In this approach, data items are mapped into high-dimensional spaces, where information about their mutual positions (in the form of inner products) is used for constructing classification, regression, or clustering rules. Kernel-based algorithms exploit the information encoded in the inner product between all pairs of data items and are successful in part because there is often an efficient method to compute inner products between very complex or even infinite dimensional vectors. Thus, kernel-based algorithms provide a way to deal with nonlinear structure by reducing nonlinear algorithms to algorithms that are linear in some feature space \mathcal{F} that is nonlinearly related to the original input space.

More precisely, assume that the data consists of vectors $X^{(1)}, \dots, X^{(n)} \in \mathcal{X} \subset \mathbb{R}^m$ and let $X \in \mathbb{R}^{m \times n}$ be the matrix whose i -th column is $X^{(i)}$. In kernel-based methods, a set of features is chosen that define a space \mathcal{F} , where it is hoped relevant structure will be revealed, the data \mathcal{X} are then mapped to the feature space \mathcal{F} using a mapping $\Phi : \mathcal{X} \rightarrow \mathcal{F}$, and then classification, regression, or clustering is performed in \mathcal{F} using traditional methods such as linear SVMs, GPs, or PCA. If \mathcal{F} is chosen to be a dot product space and if one defines the kernel matrix, also known as the Gram matrix, $G \in \mathbb{R}^{n \times n}$ as $G_{ij} = k(x_i, x_j) = (\Phi(x_i), \Phi(x_j))$, then any algorithm whose operations can be expressed in the input space in terms of dot products can be generalized to an algorithm which operates in the feature space by substituting a kernel function for the inner product. In practice, this means presenting the Gram matrix G in place of the input covariance matrix $X^T X$. Relatedly, using the kernel k instead of a dot product in the input space corresponds to mapping the data set into a (usually) high-dimensional dot product space \mathcal{F} by a (usually nonlinear) mapping $\Phi : \mathbb{R}^m \rightarrow \mathcal{F}$, and taking dot products there, i.e., $k(x_i, x_j) = (\Phi(x_i), \Phi(x_j))$. Note that for the commonly-used Mercer kernels, G is a symmetric positive semidefinite (SPSD) matrix.

The generality of this framework should be emphasized. For example, there has been much work recently on dimensionality reduction for nonlinear manifolds in high-dimensional spaces. See, e.g., Isomap, local linear embedding, and graph Laplacian eigenmap (Tenenbaum, de Silva, and Langford, 2000; Roweis and Saul, 2000; Belkin and Niyogi, 2003) as well as Hessian eigenmaps and semidefinite embedding (Donoho and Grimes, 2003; Weinberger, Sha, and Saul, 2004). These methods first induce a local neighborhood structure on the data and then use this local structure to find a global embedding of the manifold in a lower dimensional space. The manner in which these different algorithms use the local information to construct the global embedding is quite different,

but in Ham, Lee, Mika, and Schölkopf (2003) they are interpreted as kernel PCA applied to specially constructed Gram matrices.

This “kernel trick” has been quite successful for extracting nonlinear structure in large data sets when the features are chosen such that the structure in the data is more manifest in the feature space than in the original space. Although in many cases the features are chosen such that the Gram matrix is sparse, in which case sparse matrix computation methods may be used, in other applications the Gram matrix is dense, but is well approximated by a low-rank matrix. In this case, calculations of interest (such as the matrix inversion needed in GP prediction, the quadratic programming problem for SVMs, and the computation of the eigendecomposition of the Gram matrix) will still generally take space which is $O(n^2)$ and time which is $O(n^3)$. This is prohibitive if n , the number of data points, is large. Recent work in the learning theory community has focused on taking advantage of this low-rank structure in order to perform learning tasks of interest more efficiently. For example, in Achlioptas, McSherry, and Schölkopf (2002), several randomized methods are used in order to speed up kernel PCA. These methods have provable guarantees on the quality of their approximation and may be viewed as replacing the kernel function k by a “randomized kernel” which behaves like k in expectation. Relatedly, in Williams and Seeger (2001), uniform sampling without replacement is used to choose a small set of basis training points, from which an approximation to the Gram matrix is constructed. Although this algorithm does not come with provable performance guarantees, it may be viewed as a special case of our main algorithm, and it was shown empirically to perform well on two data sets for approximate GP classification and regression. It was also interpreted in terms of the Nyström method from integral equation theory; this method has also been applied recently in the learning theory community to approximate the solution of spectral partitioning for image and video segmentation (Fowlkes, Belongie, Chung, and Malik, 2004) and to extend the eigenfunctions of a data-dependent kernel to new data points (Bengio, Paiement, Vincent, Delalleau, Roux, and Ouimet, 2004; Lafon, 2004). Related work taking advantage of low-rank structure includes Smola and Schölkopf (2000); Fine and Scheinberg (2001); Williams and Seeger (2000); Burges (1996); Osuna, Freund, and Girosi (1997); Williams, Rasmussen, Schwaighofer, and Tresp (2002); Azar, Fiat, Karlin, McSherry, and Saia (2001).

1.2 Summary of Main Result

In this paper, we develop and analyze an algorithm to compute an easily-interpretable low-rank approximation to an $n \times n$ Gram matrix G . Our main result, the MAIN APPROXIMATION algorithm of Section 4.2, is an algorithm that, when given as input a SPSD matrix $G \in \mathbb{R}^{n \times n}$, computes a low-rank approximation to G of the form $\tilde{G}_k = CW_k^+C^T$, where $C \in \mathbb{R}^{n \times c}$ is a matrix formed by randomly choosing a small number c of columns (and thus rows) of G and $W_k \in \mathbb{R}^{c \times c}$ is the best rank- k approximation to W , the matrix formed by the intersection between those c columns of G and the corresponding c rows of G . The columns are chosen in c independent random trials (and thus with replacement) according to a judiciously-chosen and data-dependent nonuniform probability distribution. The nonuniform probability distribution will be carefully chosen and will be important for the provable bounds we obtain. Let $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the spectral norm and the Frobenius norm, respectively, and let G_k be the best rank- k approximation to G . Our main result, presented in a more precise form in Theorem 3, is that under appropriate assumptions:

$$\|G - CW_k^+C^T\|_\xi \leq \|G - G_k\|_\xi + \varepsilon \sum_{i=1}^n G_{ii}^2, \quad (1)$$

in both expectation and with high probability, for both $\xi = 2, F$, for all $k : 0 \leq k \leq \text{rank}(W)$. This approximation can be computed in $O(n)$ space and time after two passes over the data from external storage.

In addition to developing and analyzing an algorithm which provides a provably good decomposition of a Gram matrix, which may then be used to speed up kernel-based learning methods, this paper makes several contributions. First, it extends related work of Williams and Seeger (2001) involving uniform sampling to a more natural general case and provides a discussion of when that is necessary. Second, it provides rigorous proofs of sufficient conditions for the methods to be applicable for any data set and discusses when other conditions may be more appropriate. Third, it clarifies several potential misconceptions that have appeared in the literature regarding the relationship between recent work on Nyström-based kernel methods (Williams and Seeger, 2001; Williams, Rasmussen, Schwaighofer, and Tresp, 2002; Fowlkes, Belongie, Chung, and Malik, 2004) and the low-rank approximation algorithm of Frieze, Kannan, and Vempala (1998); Drineas, Kannan, and Mahoney (2004b). Finally, it extends random sampling methodology of the authors to a new application domain and it extends the ability of those methods from simply extracting linear structure of the data to extracting linear structure while respecting nonlinear structures such as the SPSP property.

1.3 Outline of the Paper

After this introduction, in Section 2 we provide a review of relevant linear algebra. Then, in Section 3 we review several aspects of the random sampling methodology of Drineas, Kannan, and Mahoney (2004a,b,c) that will be useful for the proofs in this paper; see also Drineas, Kannan, and Mahoney (2004d, 2005). In Section 4 we present our main algorithm and our main theorem, providing a brief discussion of the algorithm and a proof of the theorem. Then, in Section 5 we discuss in detail several aspects of the algorithm and its relationship to previous work, with a particular emphasis on the relationships between our main algorithm, the Nyström method of Williams and Seeger (2001); Williams, Rasmussen, Schwaighofer, and Tresp (2002); Fowlkes, Belongie, Chung, and Malik (2004), and our previous randomized SVD and CUR algorithms (Drineas, Kannan, and Mahoney, 2004b,c). Finally, in Section 6 we provide a brief conclusion.

2. Review of Relevant Linear Algebra

This section contains a review of linear algebra that will be useful throughout the paper. For more details about general linear algebra, see Golub and Loan (1989); Horn and Johnson (1985); Bhatia (1997); for more details about matrix perturbation theory, see Stewart and Sun (1990); and for more details about generalized inverses, see Nashed (1976); Ben-Israel and Greville (2003).

For a vector $x \in \mathbb{R}^n$ we let $|x| = \left(\sum_{i=1}^n |x_i|^2\right)^{1/2}$ denote its Euclidean length. For a matrix $A \in \mathbb{R}^{m \times n}$ we let $A^{(j)}$, $j = 1, \dots, n$, denote the j -th column of A as a column vector and $A_{(i)}$, $i = 1, \dots, m$, denote the i -th row of A as a row vector. We denote matrix norms by $\|A\|_\xi$, using subscripts to distinguish between various norms. Of particular interest will be the Frobenius norm, the square of which is $\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n A_{ij}^2$, and the spectral norm, which is defined by $\|A\|_2 = \sup_{x \in \mathbb{R}^n, x \neq 0} \frac{|Ax|}{|x|}$. These norms are related to each other as: $\|A\|_2 \leq \|A\|_F \leq \sqrt{n} \|A\|_2$. If $A \in \mathbb{R}^{m \times n}$, then there exist orthogonal matrices $U = [u^1 u^2 \dots u^m] \in \mathbb{R}^{m \times m}$ and $V = [v^1 v^2 \dots v^n] \in \mathbb{R}^{n \times n}$ where $\{u^t\}_{t=1}^m \in \mathbb{R}^m$ and

$\{v^t\}_{t=1}^n \in \mathbb{R}^n$ are such that

$$U^T A V = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_\rho),$$

where $\Sigma \in \mathbb{R}^{m \times n}$, $\rho = \min\{m, n\}$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_\rho \geq 0$. Equivalently, $A = U \Sigma V^T$. The three matrices U , V , and Σ constitute the singular value decomposition (SVD) of A . If $k \leq r = \text{rank}(A)$ and we define $A_k = U_k \Sigma_k V_k^T = \sum_{t=1}^k \sigma_t u^t v^{tT}$ then the distance (as measured by both $\|\cdot\|_2$ and $\|\cdot\|_F$) between A and any rank k approximation to A is minimized by A_k . An $n \times n$ matrix A is a symmetric positive semidefinite (SPSD) matrix if A is symmetric and $x^T A x \geq 0$ for all nonzero vectors x . If A is a SPSP matrix, then its SVD may be written $A = U \Sigma U^T$.

From the perturbation theory of matrices it is known that the size of the difference between two matrices can be used to bound the difference between the singular value spectrum of the two matrices (Stewart and Sun, 1990; Bhatia, 1997). In particular, if $A, E \in \mathbb{R}^{m \times n}$, $m \geq n$, then

$$\max_{t:1 \leq t \leq n} |\sigma_t(A+E) - \sigma_t(A)| \leq \|E\|_2 \quad (2)$$

and

$$\sum_{k=1}^n (\sigma_k(A+E) - \sigma_k(A))^2 \leq \|E\|_F^2. \quad (3)$$

The latter inequality is known as the Hoffman-Wielandt inequality.

Let $A \in \mathbb{R}^{m \times n}$, let $W \in \mathbb{R}^{m \times m}$ and $Q \in \mathbb{R}^{n \times n}$ be symmetric positive definite matrices, and consider the following generalization of the four Moore-Penrose conditions:

$$AXA = A \quad (4)$$

$$XAX = X \quad (5)$$

$$(WAX)^T = WAX \quad (6)$$

$$(QXA)^T = QXA. \quad (7)$$

The unique X that satisfies these four conditions is denoted $X = A_{(W,Q)}^{(1,2)} = A_{(W,Q)}^+$ and is the $\{W, Q\}$ -weighted- $\{1, 2\}$ -generalized inverse of A . It can be expressed in terms of the unweighted generalized inverse of A as: $A_{(W,Q)}^+ = Q^{-1/2} (W^{1/2} A Q^{-1/2})^+ W^{1/2}$. Note that if $W = I_m$ and $Q = I_n$ then the unique $X \in \mathbb{R}^{n \times m}$ satisfying these four conditions is the Moore-Penrose generalized inverse A^+ . If $r = \text{rank}(A)$, then in terms of the SVD the generalized inverse takes the following form: $A^+ = V \Sigma^{-1} U^T = \sum_{t=1}^r \sigma_t^{-1} v^t u^{tT}$.

3. Review of Our Random Sampling Methodology

Recent work in the theory of randomized algorithms has focused on matrix problems (Frieze, Kannan, and Vempala, 1998; Drineas, Frieze, Kannan, Vempala, and Vinay, 1999; Achlioptas and McSherry, 2001; Achlioptas, McSherry, and Schölkopf, 2002; Drineas and Kannan, 2001, 2003; Drineas, Kannan, and Mahoney, 2004a,b,c,d, 2005; Rademacher, Vempala, and Wang, 2005). In particular, our previous work has applied random sampling methods to the approximation of several common matrix computations such as matrix multiplication (Drineas, Kannan, and Mahoney, 2004a), the computation of low-rank approximations to a matrix (Drineas, Kannan, and Mahoney, 2004b), the computation of the CUR matrix decomposition (Drineas, Kannan, and Mahoney, 2004c), and approximating the feasibility of linear programs (Drineas, Kannan, and Mahoney, 2004d, 2005). In this section, we review two results that will be used in this paper.

3.1 Review of Approximate Matrix Multiplication

The BASICMATRIXMULTIPLICATION algorithm to approximate the product of two matrices is presented and analyzed in Drineas, Kannan, and Mahoney (2004a). When this algorithm is given as input a matrix, $A \in \mathbb{R}^{m \times n}$, a probability distribution $\{p_i\}_{i=1}^n$, and a number $c \leq n$, it returns as output a matrix $C \in \mathbb{R}^{m \times c}$ (such that $CC^T \approx AA^T$) whose columns are c randomly-chosen and suitably-rescaled columns of A . An important aspect of this algorithm is the probability distribution $\{p_i\}_{i=1}^n$ used to choose columns of A . Although one could always use a uniform distribution to choose the columns to form the matrix C , superior results are obtained if the probabilities are chosen judiciously. Sampling probabilities of the form (8), that depend on the lengths squared of the columns of A , are the *optimal sampling probabilities* for approximating AA^T by CC^T , in a sense made precise in Drineas, Kannan, and Mahoney (2004a). Note that if these probabilities are relaxed such that $p_k \geq \beta |A^{(k)}|^2 / \|A\|_F^2$ for some positive $\beta \leq 1$, then bounds similar to those in the following theorem will be obtained, with a small β -dependent loss in accuracy. Note also that although we studied random sampling with replacement for ease of analysis, it is not known how to compute efficiently optimal nonuniform sampling probabilities when the sampling is performed without replacement. In Drineas, Kannan, and Mahoney (2004a) we prove a more general version of the following theorem; see Drineas, Kannan, and Mahoney (2004a) for a discussion of the technical issues associated with this result.

Theorem 1 *Suppose $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{Z}^+$ such that $1 \leq c \leq n$, and $\{p_i\}_{i=1}^n$ are such that*

$$p_k = \frac{|A^{(k)}|^2}{\|A\|_F^2}. \tag{8}$$

Construct C with the BASICMATRIXMULTIPLICATION algorithm of Drineas, Kannan, and Mahoney (2004a), and let CC^T be an approximation to AA^T . Then,

$$\mathbf{E} [\|AA^T - CC^T\|_F] \leq \frac{1}{\sqrt{c}} \|A\|_F^2. \tag{9}$$

Furthermore, let $\delta \in (0, 1)$ and $\eta = 1 + \sqrt{8 \log(1/\delta)}$. Then, with probability at least $1 - \delta$,

$$\|AA^T - CC^T\|_F \leq \frac{\eta}{\sqrt{c}} \|A\|_F^2. \tag{10}$$

3.2 Review of Approximate Singular Value Decomposition

The LINEARTIMESVD algorithm is presented in Drineas, Kannan, and Mahoney (2004b). It is an algorithm which, when given a matrix $A \in \mathbb{R}^{m \times n}$, uses $O(m+n)$ additional space and time to compute an approximation to the top k singular values and the corresponding left singular vectors of A . It does so by randomly choosing c columns of A and rescaling each appropriately to construct a matrix $C \in \mathbb{R}^{m \times c}$, computing the top k singular values and corresponding right singular vectors of C by performing an eigendecomposition of $C^T C$, and using this information to construct a matrix $H_k \in \mathbb{R}^{m \times k}$ consisting of approximations to the top k left singular vectors of A . A minor modification of the result from Drineas, Kannan, and Mahoney (2004b) yields the following theorem in which the additional error is stated with respect to the best rank k approximation for any $k \leq \text{rank}(C)$. This theorem holds for any set of sampling probabilities, but the best bounds are obtained when probabilities of the form (8) are used, in which case Theorem 2 may be combined with Theorem 1.

Theorem 2 Suppose $A \in \mathbb{R}^{m \times n}$ and let H_k be the $m \times k$ matrix whose columns consist of the top k singular vectors of the $m \times c$ matrix C , as constructed from the LINEARTIMESVD algorithm of Drineas, Kannan, and Mahoney (2004b). Then, for every $k : 0 \leq k \leq \text{rank}(C)$,

$$\|A - H_k H_k^T A\|_F^2 \leq \|A - A_k\|_F^2 + 2\sqrt{k} \|AA^T - CC^T\|_F \quad (11)$$

$$\|A - H_k H_k^T A\|_2^2 \leq \|A - A_k\|_2^2 + 2 \|AA^T - CC^T\|_2. \quad (12)$$

In addition, if $k = r = \text{rank}(C)$ then

$$\|A - H_r H_r^T A\|_2^2 \leq \|AA^T - CC^T\|_2. \quad (13)$$

4. Approximating a Gram Matrix

Consider a set of n points in \mathbb{R}^m , denoted by $X^{(1)}, \dots, X^{(n)}$, and let X be the $m \times n$ matrix whose i -th column is $X^{(i)}$. These points may be either the original data or the data after they have been mapped into the feature space. Then, define the $n \times n$ Gram matrix G as $G = X^T X$. Thus, G is a SPSD matrix and $G_{ij} = (X^{(i)}, X^{(j)})$ is the dot product between the data vectors $X^{(i)}$ and $X^{(j)}$. If G is dense but has good linear structure, i.e., is well-approximated by a low-rank matrix, then a computation of a easily-computable and easily-interpretable low-rank approximation to G , with provable error bounds, is of interest.

In this section, two algorithms are presented that compute such an approximation to a Gram matrix G . In Section 4.1, a preliminary algorithm is presented; it is a modification of an algorithm in the literature and is a special case of our main algorithm. Then, in Section 4.2, our main algorithm and our main theorem are presented. Finally, in Section 4.3, the proof of our main theorem is presented.

4.1 A Preliminary Nyström-Based Algorithm

In Williams and Seeger (2001), a method to approximate G was proposed that, in our notation, chooses c columns from G uniformly at random and without replacement, and constructs an approximation of the form $\tilde{G} = CW^{-1}C^T$, where the $n \times c$ matrix C consists of the c chosen columns and W is a matrix consisting of the intersection of those c columns with the corresponding c rows. Analysis of this algorithm and issues such as the existence of the inverse were not addressed in Williams and Seeger (2001), but computational experiments were performed and the procedure was shown to work well empirically on two data sets (Williams and Seeger, 2001). This method has been referred to as the Nyström method (Williams and Seeger, 2001; Williams, Rasmussen, Schwaighofer, and Tresp, 2002; Fowlkes, Belongie, Chung, and Malik, 2004) since it has an interpretation in terms of the Nyström technique for solving linear integral equations (Delves and Mohamed, 1985). See Section 5 for a full discussion.

In Algorithm 1, the PRELIMINARY APPROXIMATION algorithm is presented. It is an algorithm that takes as input an $n \times n$ Gram matrix G and returns as output an approximate decomposition of the form $\tilde{G} = CW^+C^T$, where C and W are as in Williams and Seeger (2001), and where W^+ is the Moore-Penrose generalized inverse of W . The c columns are chosen uniformly at random and with replacement. Thus, the PRELIMINARY APPROXIMATION algorithm is quite similar to the algorithm of Williams and Seeger (2001), except that we sample with replacement and that we do not assume the existence of W^{-1} . Rather than analyzing this algorithm (which could be done by combining

the analysis of Section 4.3 with the uniform sampling bounds of Drineas, Kannan, and Mahoney (2004a)), we present and analyze a more general form of it, for which we can obtain improved bounds, in Section 4.2. Note, however, that if the uniform sampling probabilities are nearly optimal, in the sense that $1/n \geq \beta G_{ii}^2 / \sum_{i=1}^n G_{ii}^2$ for some positive $\beta \leq 1$ and for every $i = 1, \dots, n$, then bounds similar to those in Theorem 3 will be obtained for this algorithm, with a small β -dependent loss in accuracy.

Data : $n \times n$ Gram matrix G and $c \leq n$.

Result : $n \times n$ matrix \tilde{G} .

- Pick c columns of G in i.i.d. trials, uniformly at random with replacement; let I be the set of indices of the sampled columns.
- Let C be the $n \times c$ matrix containing the sampled columns.
- Let W be the $c \times c$ submatrix of G whose entries are $G_{ij}, i \in I, j \in I$.
- Return $\tilde{G} = CW^+C^T$.

Algorithm 1: The PRELIMINARY APPROXIMATION algorithm.

4.2 The Main Algorithm and the Main Theorem

In previous work (Drineas, Kannan, and Mahoney, 2004a,b,c,d, 2005), we showed the importance of sampling columns and/or rows of a matrix with carefully chosen nonuniform probability distributions in order to obtain provable error bounds for a variety of common matrix operations. In Algorithm 2, the MAIN APPROXIMATION algorithm is presented. It is a generalization of the PRELIMINARY APPROXIMATION algorithm that allows the column sample to be formed using arbitrary sampling probabilities. The MAIN APPROXIMATION algorithm takes as input an $n \times n$ Gram matrix G , a probability distribution $\{p_i\}_{i=1}^n$, a number $c \leq n$ of columns to choose, and a rank parameter $k \leq c$. It returns as output an approximate decomposition of the form $\tilde{G}_k = CW_k^+C^T$, where C is an $n \times c$ matrix consisting of the chosen columns of G , each rescaled in an appropriate manner, and where W_k is a $c \times c$ matrix that is the best rank- k approximation to the matrix W , which is a matrix whose elements consist of those elements in G in the intersection of the chosen columns and the corresponding rows, each rescaled in an appropriate manner.

To implement this algorithm, two passes over the Gram matrix G from external storage and $O(n)$, i.e. sublinear in $O(n^2)$, additional space and time are sufficient (assuming that the sampling probabilities of the form, e.g., $p_i = G_{ii}^2 / \sum_{i=1}^n G_{ii}^2$ or $p_i = |G^{(i)}|^2 / \|G\|_F^2$ or $p_i = 1/n$ are used). Thus, this algorithm is efficient within the framework of the Pass-Efficient model; see Drineas, Kannan, and Mahoney (2004a) for more details. Note that if the sampling probabilities of the form $p_i = G_{ii}^2 / \sum_{i=1}^n G_{ii}^2$ are used, as in Theorem 3 below, then one may store the $m \times n$ data matrix X in external storage, in which case only those elements of G that are used in the approximation need to be computed.

In the simplest application of this algorithm, one could choose $k = c$, in which case $W_k = W$, and the decomposition is of the form $\tilde{G} = CW^+C^T$, where W^+ is the exact Moore-Penrose generalized inverse of the matrix W . In certain cases, however, computing the generalized inverse may be problematic since, e.g., it may amplify noise present in the low singular values. Note that, as a function of increasing k , the Frobenius norm bound (11) of Theorem 2 is not necessarily optimal for

Data : $n \times n$ Gram matrix G , $\{p_i\}_{i=1}^n$ such that $\sum_{i=1}^n p_i = 1$, $c \leq n$, and $k \leq c$.

Result : $n \times n$ matrix \tilde{G} .

- Pick c columns of G in i.i.d. trials, with replacement and with respect to the probabilities $\{p_i\}_{i=1}^n$; let I be the set of indices of the sampled columns.
- Scale each sampled column (whose index is $i \in I$) by dividing its elements by $\sqrt{cp_i}$; let C be the $n \times c$ matrix containing the sampled columns rescaled in this manner.
- Let W be the $c \times c$ submatrix of G whose entries are $G_{ij}/(c\sqrt{p_i p_j})$, $i \in I, j \in I$.
- Compute W_k , the best rank- k approximation to W .
- Return $\tilde{G}_k = CW_k^+ C^T$.

Algorithm 2: The MAIN APPROXIMATION algorithm.

$k = \text{rank}(C)$. Also, although the bounds of Theorem 3 for the spectral norm for $k \leq \text{rank}(W)$ are in general worse than those for $k = \text{rank}(W)$, the former are of interest since our algorithms hold for any input Gram matrix and we make no assumptions about a model for the noise in the data.

The sampling matrix formalism of Drineas, Kannan, and Mahoney (2004a) is used in the proofs of Theorem 3 in Section 4.3, and thus we introduce it here. Let us define the sampling matrix $S \in \mathbb{R}^{n \times c}$ to be the zero-one matrix where $S_{ij} = 1$ if the i -th column of A is chosen in the j -th independent random trial and $S_{ij} = 0$ otherwise. Similarly, define the rescaling matrix $D \in \mathbb{R}^{c \times c}$ to be the diagonal matrix with $D_{tt} = 1/\sqrt{cp_{i_t}}$. Then the $n \times c$ matrix

$$C = GSD$$

consists of the chosen columns of G , each of which has been rescaled by $1/\sqrt{cp_{i_t}}$, where i_t is the label of the column chosen in the t -th independent trial. Similarly, the $c \times c$ matrix

$$W = (SD)^T GSD = DS^T GSD$$

consists of the intersection between the chosen columns and the corresponding rows, each element of which has been rescaled by with $1/c\sqrt{p_i p_j}$. (This can also be viewed as forming W by sampling a number c of rows of C and rescaling. Note, however, that in this case the columns of A and the rows of C are sampled using the same probabilities.) In Algorithm 3, the MAIN APPROXIMATION is restated using this sampling matrix formalism. It should be clear that Algorithm 3 and Algorithm 2 yield identical results.

Before stating our main theorem, we wish to emphasize the structural simplicity of our main result. If, e.g., we choose $k = c$, then our main algorithm provides a decomposition of the form $\tilde{G} = CW^+ C^T$:

$$\begin{pmatrix} G \end{pmatrix} \approx \begin{pmatrix} \tilde{G} \end{pmatrix} = \begin{pmatrix} C \end{pmatrix} \begin{pmatrix} W \end{pmatrix}^+ \begin{pmatrix} C^T \end{pmatrix}. \quad (14)$$

Up to rescaling, the MAIN APPROXIMATION algorithm returns an approximation \tilde{G} which is created from two submatrices of G , namely C and W . In the uniform sampling case, $p_i = 1/n$, the diagonal elements of the rescaling matrix D are all n/c , and these all cancel out of the expression. In the

Data : $n \times n$ Gram matrix G , $\{p_i\}_{i=1}^n$ such that $\sum_{i=1}^n p_i = 1$, $c \leq n$, and $k \leq c$.

Result : $n \times n$ matrix \tilde{G} .

- Define the $(n \times c)$ matrix $S = \mathbf{0}_{n \times c}$;
- Define the $(c \times c)$ matrix $D = \mathbf{0}_{c \times c}$;
- **for** $t = 1, \dots, c$ **do**
 - Pick $i_t \in [n]$, where $\Pr(i_t = i) = p_i$;
 - $D_{tt} = (cp_{i_t})^{-1/2}$;
 - $S_{i_t t} = 1$;
- **end**
- Let $C = GSD$ and $W = DS^T GSD$.
- Compute W_k , the best rank- k approximation to W .
- Return $\tilde{G}_k = CW_k^+ C^T$.

Algorithm 3: The MAIN APPROXIMATION algorithm, restated.

nonuniform sampling case, C is a rescaled version of the columns of G and W is a rescaled version of the intersection of those columns with the corresponding rows. Alternatively, one can view C as consisting of the actual columns of G , without rescaling, and W as consisting of the intersection of those columns with the corresponding rows, again without rescaling, in the following manner. Let $\hat{C} = GS$, let $\hat{W} = S^T GS$, and let

$$\hat{W}^+ = \hat{W}_{D^2, D^{-2}}^+ = D(D\hat{W}D)^+ D \quad (15)$$

be the $\{D^2, D^{-2}\}$ -weighted- $\{1, 2\}$ -generalized inverse of \hat{W} . Then $G \approx \tilde{G} = \hat{C}\hat{W}^+\hat{C}^T$.

The following theorem states our main result regarding the MAIN APPROXIMATION algorithm. Its proof may be found in Section 4.3.

Theorem 3 *Suppose G is an $n \times n$ SPSD matrix, let $k \leq c$ be a rank parameter, and let $\tilde{G}_k = CW_k^+ C^T$ be constructed from the MAIN APPROXIMATION algorithm of Algorithm 2 by sampling c columns of G with probabilities $\{p_i\}_{i=1}^n$ such that*

$$p_i = G_{ii}^2 / \sum_{i=1}^n G_{ii}^2. \quad (16)$$

Let $r = \text{rank}(W)$ and let G_k be the best rank- k approximation to G . In addition, let $\varepsilon > 0$ and $\eta = 1 + \sqrt{8 \log(1/\delta)}$. If $c \geq 64k/\varepsilon^4$, then

$$\mathbf{E} [\|G - \tilde{G}_k\|_F] \leq \|G - G_k\|_F + \varepsilon \sum_{i=1}^n G_{ii}^2 \quad (17)$$

and if $c \geq 64k\eta^2/\varepsilon^4$ then with probability at least $1 - \delta$

$$\|G - \tilde{G}_k\|_F \leq \|G - G_k\|_F + \varepsilon \sum_{i=1}^n G_{ii}^2. \quad (18)$$

In addition, if $c \geq 4/\varepsilon^2$ then

$$\mathbf{E} [\|G - \tilde{G}_k\|_2] \leq \|G - G_k\|_2 + \varepsilon \sum_{i=1}^n G_{ii}^2 \quad (19)$$

and if $c \geq 4\eta^2/\varepsilon^2$ then with probability at least $1 - \delta$

$$\|G - \tilde{G}_k\|_2 \leq \|G - G_k\|_2 + \varepsilon \sum_{i=1}^n G_{ii}^2. \quad (20)$$

Several things should be noted about this result. First, if $k \geq r = \text{rank}(W)$ then $W_k = W$, and an application of (13) of Theorem 2 leads to bounds of the form $\|G - \tilde{G}_r\|_2 \leq \varepsilon \sum_{i=1}^n G_{ii}^2$, in expectation and with high probability. Second, the sampling probabilities used in Theorem 3 may be written as $p_i = |X^{(i)}|^2 / \|X\|_F^2$, which only depend on dot products from the data matrix X . This is useful if X consists of the data after it has been mapped to the feature space \mathcal{F} . Finally, if the sampling probabilities were of the form $p_i = |G^{(i)}|^2 / \|G\|_F^2$ then they would preferentially choose data points that are more informative (in the sense of being longer) and/or more representative of the data (in the sense that they tend to be more well correlated with more data points). Instead the probabilities (16) ignore the correlations. As discussed in Sections 5 and 6, this leads to somewhat worse error bounds. To the best of our knowledge, it is not known how to sample with respect to correlations while respecting the SPSD property and obtaining provably good bounds with improved error bounds. This is of interest since in many applications it is likely that the data are approximately normalized by the way the data are generated, and it is the correlations that are of interest. Intuitively, this difficulty arises since it is difficult to identify structure in a matrix to ensure the SPSD property, unless, e.g., the matrix is diagonally dominant or given in the form $X^T X$. As will be seen in Section 4.3, the proof of Theorem 3 depends crucially on the decomposition of G as $G = X^T X$.

4.3 Proof of Theorem 3

Since $G = X^T X$ it follows that both the left and the right singular vectors of G are equal to the right singular vectors of X and that the singular values of G are the squares of the singular values of X . More formally, let the SVD of X be $X = U\Sigma V^T$. Then

$$G = V\Sigma^2 V^T = X^T U U^T X. \quad (21)$$

Now, let us consider $C_X = XSD \in \mathbb{R}^{m \times c}$, i.e., the column sampled and rescaled version of X , and let the SVD of C_X be $C_X = \hat{U}\hat{\Sigma}\hat{V}^T$. Thus, in particular, \hat{U} contains the left singular vectors of C_X . We do not specify the dimensions of \hat{U} (and in particular how many columns \hat{U} has) since we do not know the rank of C_X . Let \hat{U}_k be the $m \times k$ matrix whose columns consist of the singular vectors of C_X corresponding to the top k singular values. Instead of exactly computing the left singular vectors U of X , we can approximate them by \hat{U}_k , computed from a column sample of X , and use this to compute an approximation \tilde{G} to G .

We first establish the following lemma, which provides a bound on $\|G - \tilde{G}_k\|_\xi$ for $\xi = 2, F$.

Lemma 4 *If $\tilde{G}_k = CW_k^+ C^T$ then*

$$\|G - \tilde{G}_k\|_F = \|X^T X - X^T \hat{U}_k \hat{U}_k^T X\|_F \quad (22)$$

$$\|G - \tilde{G}_k\|_2 = \|X - \hat{U}_k \hat{U}_k^T X\|_2^2. \quad (23)$$

Proof Recall that $C = GSD$ and $W = (SD)^T GSD = C_X^T C_X$. Thus, $W = \hat{V} \hat{\Sigma}^2 \hat{V}$ and $W_k = \hat{V} \hat{\Sigma}_k^2 \hat{V}^T$, where $\hat{\Sigma}_k$ is the diagonal matrix with the top k singular values of C_X on the diagonal and the remainder set to 0. Then since $C_X = XSD = \hat{U} \hat{\Sigma} \hat{V}^T$ and $W_k^+ = \hat{V} \hat{\Sigma}_k^{-2} \hat{V}^T$

$$\tilde{G}_k = GSD(W_k)^+ (GSD)^T \quad (24)$$

$$= X^T \hat{U} \hat{\Sigma} \hat{V}^T (\hat{V} \hat{\Sigma}_k^2 \hat{V}^T)^+ \hat{V} \hat{\Sigma} \hat{U}^T X \quad (25)$$

$$= X^T \hat{U}_k \hat{U}_k^T X, \quad (26)$$

where $\hat{U}_k \hat{U}_k^T$ is a projection onto the space spanned by the top k singular vectors of W . (22) then follows immediately, and (23) follows since

$$X^T X - X^T \hat{U}_k \hat{U}_k^T X = (X - \hat{U}_k \hat{U}_k^T X)^T (X - \hat{U}_k \hat{U}_k^T X)$$

and since $\|\Omega\|_2^2 = \|\Omega^T \Omega\|_2$ for any matrix Ω . ■

By combining (23) with Theorem 2, we see that

$$\begin{aligned} \|G - \tilde{G}_k\|_2 &\leq \|X - X_k\|_2^2 + 2 \|XX^T - C_X C_X^T\|_2 \\ &\leq \|G - G_k\|_2 + 2 \|XX^T - C_X C_X^T\|_2. \end{aligned}$$

Since the sampling probabilities (16) are of the form $p_i = |X^{(i)}|^2 / \|X\|_F^2$, this may be combined with Theorem 1, from which, by choosing c appropriately, the spectral norm bounds (19) and (20) of Theorem 3 follow.

To establish the Frobenius norm bounds, define $E = XX^T XX^T - C_X C_X^T C_X C_X^T$. Then, we have that

$$\|G - \tilde{G}_k\|_F^2 = \|X^T X\|_F^2 - 2 \|XX^T \hat{U}_k\|_F^2 + \|\hat{U}_k^T XX^T \hat{U}_k\|_F^2 \quad (27)$$

$$\leq \|X^T X\|_F^2 - 2 \left(\sum_{t=1}^k \sigma_t^4(C_X) - \sqrt{k} \|E\|_F \right) + \sum_{t=1}^k \sigma_t^4(C_X) + \sqrt{k} \|E\|_F \quad (28)$$

$$= \|X^T X\|_F^2 - \sum_{t=1}^k \sigma_t^4(C_X) + 3\sqrt{k} \|E\|_F \quad (29)$$

$$\leq \|X^T X\|_F^2 - \sum_{t=1}^k \sigma_t^2(X^T X) + 4\sqrt{k} \|E\|_F, \quad (30)$$

where (27) follows by Lemmas 4 and 5, (28) follows by Lemmas 6 and 7, and (30) follows by Lemma 8. Since

$$\|X^T X\|_F^2 - \sum_{t=1}^k \sigma_t^2(X^T X) = \|G\|_F^2 - \sum_{t=1}^k \sigma_t^2(G) = \|G - G_k\|_F^2,$$

it follows that

$$\|G - \tilde{G}_k\|_F^2 \leq \|G - G_k\|_F^2 + 4\sqrt{k} \|XX^T XX^T - C_X C_X^T C_X C_X^T\|_F. \quad (31)$$

Since the sampling probabilities (16) are of the form $p_i = |X^{(i)}|^2 / \|X\|_F^2$, this may be combined with Lemma 9 and Theorem 1. Since $(\alpha^2 + \beta^2)^{1/2} \leq \alpha + \beta$ for $\alpha, \beta \geq 0$, by using Jensen's inequality, and by choosing c appropriately, the Frobenius norm bounds (17) and (18) of Theorem 3 follow.

The next four lemmas are used to bound the right hand side of (22).

Lemma 5 *For every $k : 0 \leq k \leq \text{rank}(W)$ we have that*

$$\|X^T X - X^T \hat{U}_k \hat{U}_k^T X\|_F^2 = \|X^T X\|_F^2 - 2\|XX^T \hat{U}_k\|_F^2 + \|\hat{U}_k^T XX^T \hat{U}_k\|_F^2.$$

Proof Define $Y = X - \hat{U}_k \hat{U}_k^T X$. Then

$$\begin{aligned} \|X^T X - X^T \hat{U}_k \hat{U}_k^T X\|_F^2 &= \|Y^T Y\|_F^2 \\ &= \mathbf{Tr}(Y^T Y Y^T Y) \\ &= \|X^T X\|_F^2 - 2\mathbf{Tr}(XX^T \hat{U}_k \hat{U}_k^T XX^T) + \mathbf{Tr}(\hat{U}_k^T XX^T \hat{U}_k \hat{U}_k^T XX^T \hat{U}_k), \end{aligned}$$

where the last line follows by multiplying out terms and since the trace is symmetric under cyclic permutations. The lemma follows since $\|\Omega\|_F^2 = \mathbf{Tr}(\Omega\Omega^T)$ for any matrix Ω . ■

Lemma 6 *For every $k : 0 \leq k \leq \text{rank}(W)$ we have that*

$$\left| \|XX^T \hat{U}_k\|_F^2 - \sum_{t=1}^k \sigma_t^4(C_X) \right| \leq \sqrt{k} \|XX^T XX^T - C_X C_X^T C_X C_X^T\|_F.$$

Proof Since $\sigma_t(C_X C_X^T) = \sigma_t^2(C_X)$ and since \hat{U} is a matrix consisting of the singular vectors of $C_X = XSD$, we have that

$$\begin{aligned} \left| \|XX^T \hat{U}_k\|_F^2 - \sum_{t=1}^k \sigma_t^4(C_X) \right| &= \left| \sum_{t=1}^k |XX^T \hat{U}^{(t)}|^2 - \sum_{t=1}^k |C_X C_X^T \hat{U}^{(t)}|^2 \right| \\ &= \left| \sum_{t=1}^k \hat{U}^{(t)T} (XX^T XX^T - C_X C_X^T C_X C_X^T) \hat{U}^{(t)} \right| \\ &\leq \sqrt{k} \left(\sum_{t=1}^k \left(\hat{U}^{(t)T} (XX^T XX^T - C_X C_X^T C_X C_X^T) \hat{U}^{(t)} \right)^2 \right)^{1/2}, \end{aligned}$$

where the last line follows from the Cauchy-Schwartz inequality. The lemma then follows. ■

Lemma 7 *For every $k : 0 \leq k \leq \text{rank}(W)$ we have that*

$$\|\hat{U}_k^T XX^T \hat{U}_k\|_F^2 - \sum_{t=1}^k \sigma_t^4(C_X) \leq \sqrt{k} \|XX^T XX^T - C_X C_X^T C_X C_X^T\|_F.$$

Proof Recall that if a matrix U has orthonormal columns then $\|U^T \Omega\|_F \leq \|\Omega\|_F$ for any matrix Ω . Thus, we have that

$$\begin{aligned} \|\hat{U}_k^T X X^T \hat{U}_k\|_F^2 - \sum_{t=1}^k \sigma_t^4(C_X) &\leq \|X X^T \hat{U}_k\|_F^2 - \sum_{t=1}^k \sigma_t^4(C_X) \\ &\leq \left| \|X X^T \hat{U}_k\|_F^2 - \sum_{t=1}^k \sigma_t^4(C_X) \right|. \end{aligned}$$

The remainder of the proof follows that of Lemma 6. ■

Lemma 8 For every $k : 0 \leq k \leq \text{rank}(W)$ we have that

$$\left| \sum_{t=1}^k \sigma_t^4(C_X) - \sigma_t^2(X^T X) \right| \leq \sqrt{k} \|X X^T X X^T - C_X C_X^T C_X C_X^T\|_F.$$

Proof

$$\begin{aligned} \left| \sum_{t=1}^k \sigma_t^4(C_X) - \sigma_t^2(X^T X) \right| &\leq \sqrt{k} \left(\sum_{t=1}^k (\sigma_t^4(C_X) - \sigma_t^2(X^T X))^2 \right)^{1/2} \\ &= \sqrt{k} \left(\sum_{t=1}^k (\sigma_t(C_X C_X^T C_X C_X^T) - \sigma_t(X X^T X X^T))^2 \right)^{1/2} \\ &\leq \sqrt{k} \|X X^T X X^T - C_X C_X^T C_X C_X^T\|_F, \end{aligned}$$

where the first inequality follows from the Cauchy-Schwartz inequality and the second inequality follows from the matrix perturbation result (3). ■

The following is a result of the BASICMATRIXMULTIPLICATION algorithm that is not found in Drineas, Kannan, and Mahoney (2004a), but that will be useful for bounding the additional error in (31). We state this result for a general $m \times n$ matrix A .

Lemma 9 Suppose $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{Z}^+$ such that $1 \leq c \leq n$, and $\{p_i\}_{i=1}^n$ are such that $p_k = |A^{(k)}|^2 / \|A\|_F^2$. Construct C with the BASICMATRIXMULTIPLICATION algorithm of Drineas, Kannan, and Mahoney (2004a). Then

$$\mathbf{E} [\|A A^T A A^T - C C^T C C^T\|_F] \leq \frac{2}{\sqrt{c}} \|A\|_F^4. \quad (32)$$

Furthermore, let $\delta \in (0, 1)$ and $\eta = 1 + \sqrt{8 \log(1/\delta)}$. Then, with probability at least $1 - \delta$,

$$\|A A^T A A^T - C C^T C C^T\|_F \leq \frac{2\eta}{\sqrt{c}} \|A\|_F^4. \quad (33)$$

Proof First note that:

$$\begin{aligned} AA^T AA^T - CC^T CC^T &= AA^T AA^T - AA^T CC^T + AA^T CC^T - CC^T CC^T \\ &= AA^T (AA^T - CC^T) + (AA^T - CC^T) CC^T. \end{aligned}$$

Thus, by submultiplicativity and subadditivity we have that for $\xi = 2, F$:

$$\|AA^T AA^T - CC^T CC^T\|_F \leq \|A\|_F^2 \|AA^T - CC^T\|_F + \|AA^T - CC^T\|_F \|C\|_F^2.$$

The lemma follows since $\|C\|_F^2 = \|A\|_F^2$ when $p_k = |A^{(k)}|^2 / \|A\|_F^2$, and by applying Theorem 1. ■

5. Discussion Section

One motivation for the present work was to provide a firm theoretical basis for the Nyström-based algorithm of Williams and Seeger (2001). A second motivation was to clarify the relationships between our randomized SVD algorithms (Drineas, Kannan, and Mahoney, 2004b), our randomized CUR algorithms (Drineas, Kannan, and Mahoney, 2004c), and the Nyström-based methods of others (Williams and Seeger, 2001; Williams, Rasmussen, Schwaighofer, and Tresp, 2002; Fowlkes, Belongie, Chung, and Malik, 2004). A third motivation was to extend our random sampling methodology to extract linear structure from matrices while preserving important nonlinear structure. In this section, we discuss these issues. Note that our CONSTANTTIMESVD algorithm of Drineas, Kannan, and Mahoney (2004b) is the algorithm originally analyzed by Frieze, Kannan, and Vempala (1998), and thus a discussion of it corresponds also to a discussion of their algorithm (Frieze, Kannan, and Vempala, 1998).

5.1 Summary of the Nyström Method

The Nyström method was originally introduced to handle approximations based on the numerical integration of the integral operator in integral equations, and it is well known for its simplicity and accuracy (Delves and Mohamed, 1985). To illustrate the Nyström method, consider the eigenfunction problem:

$$\int_D K(t, s) \Phi(s) ds = \lambda \Phi(t) \quad t \in D. \quad (34)$$

The resulting solution is first found at the set of quadrature node points, and then it is extended to all points in D by means of a special interpolation formula (see (39) below). This method requires the use of a quadrature rule. Assume that $D = [a, b] \subset \mathbb{R}$ and that the quadrature rule is the following:

$$\int_a^b y(s) ds = \sum_{j=1}^n w_j y(s_j), \quad (35)$$

where $\{w_j\}$ are the weights and $\{s_j\}$ are the quadrature points that are determined by the particular quadrature rule. If this rule is used to compute the integral occurring in (34), we have

$$\int_a^b K(x, s) \Phi(s) ds \approx \sum_{j=1}^n w_j k(x, s_j) \tilde{\Phi}(s_j), \quad (36)$$

and the integral equation (34) leads to an eigenvalue problem of the form

$$\sum_{j=1}^n w_j k(x, s_j) \tilde{\phi}(s_j) = \tilde{\lambda} \tilde{\phi}(x). \tag{37}$$

Solving (37) leads to an approximate eigenvalue $\tilde{\lambda}$ and an approximate eigenfunction $\tilde{\phi}(x)$ and may be done via the Nyström method as follows. First, set $x = x_i, i = 1, \dots, n$ in (37). This leads to a system of n algebraic equations:

$$\sum_{j=1}^n w_j k(x_i, s_j) \tilde{\phi}(s_j) = \tilde{\lambda} \tilde{\phi}(x_i), \tag{38}$$

that depend on the set $\{x_i\}$ of Nyström points. Although it is by no means necessary that the set of Nyström points is coincident with the set of quadrature points, they are often chosen to be so since in that case if the kernel $K(\cdot, \cdot)$ is symmetric then the matrix $k(\cdot, \cdot)$ in (38) is symmetric. Then, if $\tilde{\lambda}_m \neq 0$ the exact eigenvectors $\tilde{\phi}_m$ on the Nyström points can be extended to a function $\bar{\phi}_m(x)$ on the full domain by substituting it into (37):

$$\bar{\phi}_m(x) = \frac{1}{\tilde{\lambda}_m} \sum_{j=1}^n w_j k(x, s_j) \tilde{\phi}_m(s_j). \tag{39}$$

The function $\bar{\phi}_m(x)$ is the *Nyström extension* of the eigenvector $\tilde{\phi}_m$, and in the present context may be thought of as being an approximation to the exact eigenfunction Φ_m computed by extending a function computed on a (small) number n of points to the full (large) domain D .

In the applications we are considering, the data points are vectors in \mathbb{R}^n . Thus, consider an $m \times n$ matrix A consisting of m such vectors. Let c columns and r rows be chosen (without replacement) in some manner, and let A be partitioned as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{40}$$

where $A_{11} \in \mathbb{R}^{c \times r}$ represents the subblock of matrix elements common to the sampled columns and the sampled rows, A_{21} and A_{12} are rectangular matrices consisting of elements with a sampled column label (exclusive) or sampled row label, respectively, and $A_{22} \in \mathbb{R}^{(m-c) \times (n-r)}$ consists of the remaining elements. If $c, r = O(1)$ then A_{11} is small and A_{22} is large. To be consistent with the notation of Drineas, Kannan, and Mahoney (2004b,c), we let $C = [A_{11}^T A_{21}^T]^T$ and $R = [A_{11} A_{12}]$. Let the SVD of A_{11} be $A_{11} = \tilde{U} \tilde{\Sigma} \tilde{V}^T$, and let the rank of A_{11} be k .

Assume, for the moment, that A is a SPSD matrix and that the chosen rows are the same as the chosen columns. Then, A_{11} is also a SPSD matrix; in addition, $\tilde{V} = \tilde{U}$ are the eigenvalues of A_{11} and $\tilde{\Sigma}$ consists of the eigenvectors of A_{11} . In this case, the Nyström extension of \tilde{U} gives the following approximation for the eigenvectors of the full matrix A :

$$\bar{U} = C \tilde{U} \tilde{\Sigma}^{-1} = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} \tilde{U} \tilde{\Sigma}^{-1} = \begin{bmatrix} \tilde{U} \\ A_{21} \tilde{U} \tilde{\Sigma}^{-1} \end{bmatrix}. \tag{41}$$

Note that this Nyström extension of the restricted solution to the full set of data points is of the same form as (39).

More generally, if A is an arbitrary $m \times n$ matrix, then the Nyström extension of \tilde{U} and \tilde{V} gives the following approximation for the singular vectors of the full matrix A :

$$\bar{U} = \begin{bmatrix} \tilde{U} \\ A_{21}\tilde{V}\tilde{\Sigma}^{-1} \end{bmatrix}, \text{ and} \quad (42)$$

$$\bar{V} = \begin{bmatrix} \tilde{V} \\ A_{12}^T\tilde{U}\tilde{\Sigma}^{-1} \end{bmatrix}. \quad (43)$$

If both \bar{U} and \bar{V} have been computed then the Nyström extensions (42)–(43) also have an interpretation in terms of matrix completion. To see this, set $\tilde{A} = \bar{U}\tilde{\Sigma}\bar{V}^T$; then we have

$$\tilde{A} = \begin{bmatrix} \tilde{U} \\ A_{21}\tilde{V}\tilde{\Sigma}^{-1} \end{bmatrix} \tilde{\Sigma} \begin{bmatrix} \tilde{V}^T & \tilde{\Sigma}^{-1}\tilde{U}^T A_{12} \end{bmatrix} \quad (44)$$

$$= \begin{bmatrix} A_{11} & \tilde{U}\tilde{U}^T A_{12} \\ A_{21}\tilde{V}\tilde{V}^T & A_{21}A_{11}^+ A_{12} \end{bmatrix} \quad (45)$$

$$= \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} A_{11}^+ \begin{bmatrix} A_{11} & A_{12} \end{bmatrix}. \quad (46)$$

Note that if A_{11} is nonsingular, then (45) becomes

$$\tilde{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{21}A_{11}^{-1}A_{12} \end{bmatrix}. \quad (47)$$

In this case, the Nyström extension implicitly approximates A_{22} using $A_{21}A_{11}^{-1}A_{12}$, and the quality of the approximation of A by \tilde{A} can be quantified by the norm of the Schur complement

$$\|A_{22} - A_{21}A_{11}^{-1}A_{12}\|_{\xi}, \xi = 2, F.$$

The size of this error norm is governed, e.g., by the extent to which the columns of A_{21} provide a good basis for the columns of A_{22} . If A_{11} is rectangular or square and singular then other terms in the matrix \tilde{A} also contribute to the error. Note that (46) is of the form $A \approx \tilde{A} = CA_{11}^+R$. If A is a SPSD matrix and the chosen rows are the same as the chosen columns then (45) is modified appropriately and (46) is of the form $A \approx \tilde{A} = CW^+C^T$, which is the form of our main decomposition for a Gram matrix G . Note, however, that neither \tilde{U} nor \tilde{V} are actually computed by our main approximation algorithm. In Sections 5.2 and 5.3, we discuss these issues further.

5.2 Relationship to the Randomized Singular Value Decompositions

Recall that the LINEARTIMESVD of Drineas, Kannan, and Mahoney (2004b) computes exactly the low-dimensional singular vectors of C . Let the SVD of C be $C = H\Sigma Z^T$. Then, the high-dimensional singular vectors of C are computed by extending the low-dimensional singular vectors as

$$H = CZ\Sigma^{-1}, \quad (48)$$

and it is these that are taken as approximations of the left singular vectors of the original matrix A , in the sense that under appropriate assumptions,

$$\|A - HH^T A\|_{\xi} \leq \|A - A_k\|_{\xi} + \varepsilon \|A\|_F, \quad (49)$$

in expectation and with high probability, for both $\xi = 2, F$. This is not a Nyström extension in the sense of Section 5.1 since although sampling is used to construct the matrix C a second level of sampling is never performed to construct A_{11} .

On the other hand, the CONSTANTTIMESVD algorithm of Drineas, Kannan, and Mahoney (2004b) (and thus the algorithm of Frieze, Kannan, and Vempala, 1998) is similar except that it *approximates* the low-dimensional singular vectors of C . It does this by randomly sampling w rows of C and rescaling each appropriately to form a $w \times c$ matrix A_{11} (this matrix is called W in Drineas, Kannan, and Mahoney (2004b,c), but it is constructed with different sampling probabilities than the W defined in this paper) and computing the eigenvectors of $A_{11}^T A_{11}$. These eigenvectors are then Nyström-extended via (42) to vectors \bar{U} (denoted by \tilde{H} in Drineas, Kannan, and Mahoney (2004b)) that approximate the left singular vectors of A . In this case, the projection $HH^T = C(C^T C)^+ C^T$ of the LINEARTIMESVD algorithm is replaced by an approximate projection onto the column space of C of the form $\bar{U}\bar{U} = C(A_{11}^T A_{11})^+ C^T$. From this perspective, since $C^T C \approx A_{11}^T A_{11}$ we may view the LINEARTIMESVD of Drineas, Kannan, and Mahoney (2004b) as performing a Nyström-based extension of approximations of the eigenvectors of $A_{11}^T A_{11}$.

We emphasize these points since we would like to clarify several potential misunderstandings in the literature regarding the relationship between the Nyström-based algorithm of Williams and Seeger (2001) and the approximate SVD algorithm of Frieze, Kannan, and Vempala (1998). For example, in some work (Williams and Seeger, 2001; Williams, Rasmussen, Schwaighofer, and Tresp, 2002; Fowlkes, Belongie, Chung, and Malik, 2004) it is claimed that their Nyström-based methods are a special case of Frieze, Kannan, and Vempala (1998) and thus of the CONSTANTTIMESVD algorithm of Drineas, Kannan, and Mahoney (2004b). Although the SVD algorithms of Drineas, Kannan, and Mahoney (2004b) and Frieze, Kannan, and Vempala (1998) do represent a Nyström-based extension in the sense just described, several things should be noted. First, in order to obtain provable performance guarantees, the CONSTANTTIMESVD algorithm used by Drineas, Kannan, and Mahoney (2004b) and Frieze, Kannan, and Vempala (1998) approximates the left (or right, but not both) singular vectors in a single Nyström-like extension of the form (42) (or (43) for the right singular vectors). This algorithm makes no assumptions about the symmetry or positive definiteness of the input matrix, and it does not take advantage of this structure if it exists. Second, and relatedly, in this algorithm there are two levels of sampling, and only the first depends directly on the elements of the matrix A ; the second depends on the lengths of the rows of C . Thus, in general, the matrix A_{11} does not consist of the same rows as columns, even if A is a SPSD matrix. If A is a SPSD matrix, then one could approximate A as $\tilde{A} = \bar{U}\tilde{\Sigma}\bar{U}^T$, but the error associated with this is not the error that the theorems of Drineas, Kannan, and Mahoney (2004b) and Frieze, Kannan, and Vempala (1998) bound. Third, the structure of the approximation obtained by Drineas, Kannan, and Mahoney (2004b) and Frieze, Kannan, and Vempala (1998) is quite different from that of the approximation of Williams and Seeger (2001) and (14). In the latter case it is of the form $CW^+ C^T$, while in the former case it is of the form $P_C A$, where P_C is an exact or approximate projection onto the column space of C .

5.3 Relationship to the Randomized CUR Decompositions

To shed further light on the relationship between the CONSTANTTIMESVD algorithm (Drineas, Kannan, and Mahoney, 2004b; Frieze, Kannan, and Vempala, 1998) and the Nyström-based methods (Williams and Seeger, 2001; Williams, Rasmussen, Schwaighofer, and Tresp, 2002; Fowlkes,

Belongie, Chung, and Malik, 2004), it is worth considering the CUR decompositions of Drineas, Kannan, and Mahoney (2004c), which are structurally a generalization of our main matrix decomposition. A *CUR decomposition* is a low-rank matrix decomposition of the form $A \approx CUR$, where C is a matrix consisting of a small number of columns of A , R is a matrix consisting of a small number of rows of A , and U is an appropriately-defined low-dimensional matrix. Examples may be found in Drineas, Kannan, and Mahoney (2004c), and also in Goreinov, Tyrtshnikov, and Zamarashkin (1997); Goreinov and Tyrtshnikov (2001). In particular, the LINEARTIMECUR and CONSTANT-TIMECUR algorithms of Drineas, Kannan, and Mahoney (2004c) (so named due to their relationship with the correspondingly-named SVD algorithms of Drineas, Kannan, and Mahoney, 2004b) compute an approximation to a matrix $A \in \mathbb{R}^{m \times n}$ by sampling c columns and r rows of the matrix A to form matrices $C \in \mathbb{R}^{m \times c}$ and $R \in \mathbb{R}^{r \times n}$, respectively. The matrices C and R are constructed with carefully-chosen and data-dependent nonuniform probability distributions, and from C and R a matrix $U \in \mathbb{R}^{c \times r}$ is constructed such that under appropriate assumptions:

$$\|A - CUR\|_{\xi} \leq \|A - A_k\|_{\xi} + \varepsilon \|A\|_F, \quad (50)$$

with high probability, for both $\xi = 2, F$. Although these algorithms apply to any matrix, and thus to a SPSD matrix, the computed approximation CUR (with the provable error bounds of the form (50)) is neither symmetric nor positive semidefinite in the latter case. The SPSD property is an important property in many applications, and thus it is desirable to obtain a low-rank approximation that respects this property. The analysis of the MAIN APPROXIMATION algorithm shows that if G is a SPSD matrix then we can choose $R = C^T$ and $U = A_{11}^+$ and obtain a SPSD approximation of the form $G \approx \tilde{G}_k = CW_k^+ C^T$ with provable error bounds of the form (1). Note that this bound is worse than that of (50) since the scale of the additional error is larger. Although it may not be surprising that the bound is somewhat worse since we are requiring that the approximation is not just low rank but that in addition it respects the nonlinear SPSD property, the worse bound is likely due simply to the sampling probabilities that were used to obtain provable performance guarantees.

Since the CUR algorithms of Drineas, Kannan, and Mahoney (2004c) rely for their proofs of correctness on the corresponding SVD algorithms of Drineas, Kannan, and Mahoney (2004b), the Nyström discussion about the SVD algorithms is relevant to them. In addition, to understand the CUR algorithm in terms of matrix completion, consider an $m \times n$ matrix A with c columns and r rows chosen in some manner which is partitioned as in (40). Let $U \in \mathbb{R}^{c \times r}$ be an appropriately defined matrix as in Drineas, Kannan, and Mahoney (2004c), and let us decompose the original matrix A of (40) as $A \approx CUR$:

$$CUR = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} U \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} \quad (51)$$

$$= \begin{bmatrix} A_{11}UA_{11} & A_{11}UA_{12} \\ A_{21}UA_{11} & A_{21}UA_{12} \end{bmatrix}. \quad (52)$$

In Drineas, Kannan, and Mahoney (2004c) $U \neq A_{11}$, but we provide a definition for U such that $U \approx A_{11}^+$, in which case the structural similarity between (51) and (46) should be clear, as should the similarity between (52) and (45). For general matrices A , the CUR decomposition approximates A_{22} by $A_{22} = A_{21}UA_{12}$, but it also approximates A_{21} by $A_{21}UA_{11}$, A_{12} by $A_{11}UA_{12}$, and A_{11} by $A_{11}UA_{11}$. Thus, the quality of the approximation of the full matrix can not be quantified simply by the norm of the Schur complement $\|A_{22} - A_{21}A_{11}^+A_{12}\|_{\xi}$, and in Drineas, Kannan, and Mahoney (2004c) we

bound $\|A - CUR\|_{\xi}$ directly. Relatedly, the quality of the approximation is determined, e.g., by how well a basis the chosen columns of C are for the remaining columns of A .

6. Conclusion

We have presented and analyzed an algorithm that provides an approximate decomposition of an $n \times n$ Gram matrix G which is of the form $G \approx \tilde{G}_k = CW_k^+C^T$ and which has provable error bounds of the form (1). A crucial feature of this algorithm is the probability distribution used to randomly sample columns. We conclude with two open problems related to the choice of this distribution.

First, it would be desirable to choose the probabilities in Theorem 3 to be $p_i = |G^{(i)}|^2 / \|G\|_F^2$ and to establish bounds of the form (1) in which the scale of the additional error was $\|G\|_F = \|X^T X\|_F$ rather than $\sum_{i=1}^n G_{ii}^2 = \|X\|_F^2$. This would entail extracting linear structure while simultaneously respecting the SPSD property and obtaining improved scale of error. This would likely be a corollary of a CUR decomposition for a general $m \times n$ matrix A with error bounds of the form (50) in which $U = W_k^+$, where W is now the matrix consisting of the intersection of the chosen columns and (in general different) rows. This would simplify considerably the form of U found in Drineas, Kannan, and Mahoney (2004c) and would lead to improved interpretability. Second, we should also note that if capturing coarse statistics over the data is not of interest, but instead one is interested in other properties of the data, e.g., identifying outliers, then probabilities that depend on the data in some other manner, e.g., inversely with respect to their lengths squared, may be appropriate. We do not have provable bounds in this case. We should note, however, that we are empirically evaluating the applicability of the methodology presented in this paper for problems of interest in machine learning. We will report the results at a future date.

Acknowledgments

We would like to thank Ravi Kannan for many fruitful discussions and the Institute for Pure and Applied Mathematics at UCLA for its generous hospitality.

References

- D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 611–618, 2001.
- D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In *Annual Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 335–342, 2002.
- Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 619–626, 2001.
- M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.
- A. Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. Springer-Verlag, New York, 2003.

- Y. Bengio, J. F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In *Annual Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, pages 177–184, 2004.
- R. Bhatia. *Matrix Analysis*. Springer-Verlag, New York, 1997.
- C. J. C. Burges. Simplified support vector decision rules. In *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, 1996.
- N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000.
- L. M. Delves and J. L. Mohamed. *Computational Methods for Integral Equations*. Cambridge University Press, Cambridge, 1985.
- D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA*, 100(10):5591–5596, 2003.
- P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 291–299, 1999.
- P. Drineas and R. Kannan. Fast Monte-Carlo algorithms for approximate matrix multiplication. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 452–459, 2001.
- P. Drineas and R. Kannan. Pass efficient algorithms for approximating large matrices. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 2003.
- P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. Technical Report YALEU/DCS/TR-1269, Yale University Department of Computer Science, New Haven, CT, February 2004a. Accepted for publication in the *SIAM Journal on Computing*.
- P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. Technical Report YALEU/DCS/TR-1270, Yale University Department of Computer Science, New Haven, CT, February 2004b. Accepted for publication in the *SIAM Journal on Computing*.
- P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. Technical Report YALEU/DCS/TR-1271, Yale University Department of Computer Science, New Haven, CT, February 2004c. Accepted for publication in the *SIAM Journal on Computing*.
- P. Drineas, R. Kannan, and M. W. Mahoney. Sampling sub-problems of heterogeneous Max-Cut problems and approximation algorithms. Technical Report YALEU/DCS/TR-1283, Yale University Department of Computer Science, New Haven, CT, April 2004d.

- P. Drineas, R. Kannan, and M. W. Mahoney. Sampling sub-problems of heterogeneous Max-Cut problems and approximation algorithms. In *Proceedings of the 22nd Annual International Symposium on Theoretical Aspects of Computer Science*, pages 57–68, 2005.
- P. Drineas and M. W. Mahoney. Approximating a Gram matrix for improved kernel-based learning. In *Proceedings of the 18th Annual Conference on Learning Theory*, pages 323–337, 2005a.
- P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. Technical Report YALEU/DCS/TR-1319, Yale University Department of Computer Science, New Haven, CT, April 2005b.
- S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.
- C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.
- A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 370–378, 1998.
- G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989.
- S. A. Goreinov and E. E. Tyrtyshnikov. The maximum-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 280:47–51, 2001.
- S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and Its Applications*, 261:1–21, 1997.
- J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. Technical Report TR-110, Max Planck Institute for Biological Cybernetics, July 2003.
- R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, 1985.
- S. Lafon. *Diffusion Maps and Geometric Harmonics*. PhD thesis, Yale University, 2004.
- M. Z. Nashed, editor. *Generalized Inverses and Applications*. Academic Press, New York, 1976.
- E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing VII*, pages 276–285, 1997.
- L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via iterative sampling. Technical Report MIT-LCS-TR-983, Massachusetts Institute of Technology, Cambridge, MA, March 2005.
- S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290:2323–2326, 2000.

- B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
- A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 911–918, 2000.
- G. W. Stewart and J. G. Sun. *Matrix Perturbation Theory*. Academic Press, New York, 1990.
- J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.
- K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the 21st International Conference on Machine Learning*, pages 839–846, 2004.
- C. K. I. Williams, C. E. Rasmussen, A. Schwaighofer, and V. Tresp. Observations on the Nyström method for Gaussian process prediction. Technical report, University of Edinburgh, 2002.
- C. K. I. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1159–1166, 2000.
- C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Annual Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 682–688, 2001.

Expectation Consistent Approximate Inference

Manfred Opper

*ISIS, School of Electronics and Computer Science
University of Southampton
SO17 1BJ, United Kingdom*

MO@ECS.SOTON.AC.UK

Ole Winther

*Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark*

OWI@IMM.DTU.DK

Editor: Michael I. Jordan

Abstract

We propose a novel framework for approximations to intractable probabilistic models which is based on a free energy formulation. The approximation can be understood as replacing an average over the original intractable distribution with a tractable one. It requires two tractable probability distributions which are made consistent on a set of moments and encode different features of the original intractable distribution. In this way we are able to use Gaussian approximations for models with discrete or bounded variables which allow us to include non-trivial correlations. These are neglected in many other methods. We test the framework on toy benchmark problems for binary variables on fully connected graphs and 2D grids and compare with other methods, such as loopy belief propagation. Good performance is already achieved by using single nodes as tractable substructures. Significant improvements are obtained when a spanning tree is used instead.

1. Introduction

Recent developments in data acquisition and computational power have spurred an increased interest in flexible statistical Bayesian models in many areas of science and engineering. Inference in probabilistic models is in many cases intractable; the computational cost of marginalization operations can scale exponentially in the number of variables or require integrals over multivariate non-tractable distributions. In order to treat systems with a large number of variables, it is therefore necessary to use approximate polynomial complexity inference methods.

Probably the most prominent and widely developed approximation technique is the so-called *variational* (or *variational Bayes*) approximation (see, e.g. Jordan et al., 1999; Attias, 2000; Bishop et al., 2003). In this approach, the true intractable probability distribution is approximated by another one which is optimally chosen within a given, tractable family minimizing the *Kullback Leibler (KL) divergence* as the measure of dissimilarity between distributions. We will use the name *variational bound* for this specific method because the approximation results in an upper bound to the *free energy* (an entropic quantity related to the KL divergence). The alternative approximation methods discussed in this paper can also be derived from the variation of an approximate free energy which is not necessarily

a bound. The most important tractable families of distributions in the variational bound approximation are multivariate Gaussians and distributions often in the exponential family which factorize in the marginals of all or for certain disjoint groups of variables (Attias, 2000) (this is often called a mean-field approximation). The use of multivariate Gaussians makes it possible to retain a significant amount of correlation between variables in the approximation. However, their application in the variational bound approximation is limited to distributions of continuous variables which have the entire real space as their natural domain. This is due to the fact that the KL divergence would diverge for distributions with non-matching support. Hence, in a majority of those applications, where random variables with constrained values (such as Boolean ones) appear, variational distributions of the mean field type have to be chosen. However, such factorizing approximations have the drawback that correlations are neglected and one often observes that fluctuations are underestimated (MacKay, 2003; Opper and Winther, 2004).

Recently, a lot of effort has been devoted to the development of approximation techniques which give an improved performance compared to the variational bound approximation. Thomas Minka’s *Expectation Propagation* (EP) approach (Minka, 2001a,b) seems to provide a general framework from which many of these developments can be re-derived and understood. EP is based on a dynamical picture where factors—their product forming a global tractable approximate distribution—are iteratively optimized. In contrast to the variational bound approach, the optimization proceeds *locally* by minimizing KL divergences between appropriately defined *marginal* distributions. Since the resulting algorithm can be formulated in terms of the matching of marginal moments, this would not rule out factorizations where discrete distributions are approximated by multivariate Gaussians. However, such a choice seems to be highly unnatural from the derivation of the EP approximation (by the infinite KL measure) and has to our knowledge not been suggested so far (Minka, private communication). Hence, in practice, the correlations between discrete variables have been mainly treated using tree-based approximations. This includes the celebrated Bethe-Kikuchi approach (Yedidia et al., 2001; Yuille, 2002; Heskes et al., 2003), for EP interpretations see Minka (2001a,b) and Minka and Qi (2004). For a variety of related approximations within statistical physics see Suzuki (1995). However, while tree-type approximations often work well for sparsely connected graphs they become inadequate for inference problems in a dense graph regardless of the type of variables.

In this paper we present an alternative view of local-consistency approximations of the EP-type which we call *expectation consistent* (EC) approximations. It can be understood by requiring consistency between *two* complementary global approximations which may have different support (say, a Gaussian one and one that factorizes into marginals). Our method is a generalization of the *adaptive TAP* approach (ADATAP) (Opper and Winther, 2001a,b) developed for inference on densely connected graphical models. Although it has been applied successfully to a variety of problems ranging from probabilistic ICA (Hojen-Sorensen et al., 2002) over Gaussian process models (Opper and Winther, 2000) to bootstrap methods for kernel machines (Malzahn and Opper, 2003), see Appendix A, its potential as a fairly general scheme has been somewhat overlooked in the Machine Learning community.¹

1. This is probably due to the fact that the most detailed description of the method has so far only appeared in the statistical physics literature (Opper and Winther, 2001a,b) in a formulation that is not very accessible to a general audience. Shortly after the method first appeared—in the context of Gaussian

Although one algorithmic realization of our method can be given an EP-style interpretation (Csató et al., 2002), we believe that it is more natural and more powerful to base the derivation on a framework of optimizing a free energy approximation. This not only has the advantage of providing a simple and clear way for adapting the model parameters within the empirical Bayes framework, but also motivates different practical optimization algorithms among which the EP-style may not always be the best choice.

Our paper is organized as follows: Section 2 motivates approximate inference and explains the notation. The expectation consistent (EC) approximation to the free energy is derived in Section 3. Examples for EC free energies are given in Section 4. The algorithmic issues are treated in Section 5, simulations in Section 6 and finally we conclude in Section 7.

2. Motivation: Approximate Inference

We consider the problem of computing expectations, i.e. certain sums or integrals involving a probability distribution with density $p(\mathbf{x})$ for a vector of random variables $\mathbf{x} = (x_1, x_2, \dots, x_N)$. We assume that such computations are considered *intractable*, either because the necessary sums are over a too large number of variables or because multivariate integrals cannot be evaluated exactly. A further complication might occur when the density itself is expressed by a non-normalized multivariate function $f(\mathbf{x})$, say, equal to the product of a prior and a likelihood, which requires further normalization, i.e.

$$p(\mathbf{x}) = \frac{1}{Z} f(\mathbf{x}) , \tag{1}$$

where the *partition function* Z must be obtained by the (intractable) summation or integration of f : $Z = \int d\mathbf{x} f(\mathbf{x})$. In a typical scenario, $f(\mathbf{x})$ is expressed as a product of two functions

$$f(\mathbf{x}) = f_q(\mathbf{x}) f_r(\mathbf{x}) \tag{2}$$

with $f_{q,r}(\mathbf{x}) \geq 0$, where f_q is “simple” enough to allow for tractable computations. The goal is to approximate the “complicated” part $f_r(\mathbf{x})$ by replacing it with a “simpler” function, say of some exponential form

$$\exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})) \equiv \exp\left(\sum_{j=1}^K \lambda_j g_j(\mathbf{x})\right) . \tag{3}$$

We have used the same vector notation for \mathbf{g} -vectors as for the random variables \mathbf{x} , however one should note that vectors will often have different dimensionalities, i.e. $K \neq N$. The vector of functions \mathbf{g} is chosen in such a way that the desired sums or integrals can be calculated in an efficient way and the parameters $\boldsymbol{\lambda}$ are adjusted to optimize certain criteria. Hence, the word tractability should always be understood as relative to some approximating set of functions \mathbf{g} .

Our framework of approximation will be restricted to problems, where both parts f_q and f_r can be considered as tractable relative to some suitable \mathbf{g} , and the intractability

processes (Opper and Winther, 2000)–Minka introduced his EP framework and showed the equivalence of the fixed points of the two methods for Gaussian process models.

of the density p arises from forming their product.² In such a case, one may alternatively retain f_r but replace f_q by an approximation of the form eq. (3). One would then end up with two types of approximations

$$q(\mathbf{x}) = \frac{1}{Z_q(\boldsymbol{\lambda}_q)} f_q(\mathbf{x}) \exp(\boldsymbol{\lambda}_q^T \mathbf{g}(\mathbf{x})) \quad (4)$$

$$r(\mathbf{x}) = \frac{1}{Z_r(\boldsymbol{\lambda}_r)} f_r(\mathbf{x}) \exp(\boldsymbol{\lambda}_r^T \mathbf{g}(\mathbf{x})) \quad (5)$$

for the same density, where $Z_q(\boldsymbol{\lambda}_q) = \int d\mathbf{x} f_q(\mathbf{x}) \exp(\boldsymbol{\lambda}_q^T \mathbf{g}(\mathbf{x}))$. We will not assume that either choice q and r is a reasonably good approximation for the global joint density $p(\mathbf{x})$ as we do in the variational bound approximation. In fact, later we will apply our method to the case of Ising variables, where the KL divergence between one of them and p is even infinite! Though, suitable different marginalizations of q and r can give quite accurate answers for important marginal statistics.

Take, as an example, the density $p(\mathbf{x}) = f(\mathbf{x})/Z = f_q(\mathbf{x})f_r(\mathbf{x})/Z$ —with respect to the Lebesgue measure in R^N —with

$$f_q(\mathbf{x}) = \prod_i \psi_i(x_i) \quad (6)$$

$$f_r(\mathbf{x}) = \exp\left(\sum_{i<j} x_i J_{ij} x_j + \sum_i \theta_i x_i\right), \quad (7)$$

where, in order to have a nontrivial problem, ψ_i should be a non-Gaussian function. We will name this the *quadratic model*. Usually there will be an ambiguity in the choice of factorization, e.g. we could have included $\exp(\sum_i \theta_i x_i)$ as a part of $f_q(\mathbf{x})$. One may approximate $p(\mathbf{x})$ by a factorizing distribution, thereby replacing $f_r(\mathbf{x})$ by some function which factorizes in the components x_i . Alternatively, one can consider replacing $f_q(\mathbf{x})$ by a Gaussian function to make the whole distribution Gaussian. Both approximations are not ideal. The first completely neglects correlations of the variables but leads to marginal distributions of the x_i , which might qualitatively resemble the non-Gaussian shape of the true marginal. The second one neglects the non-Gaussian effects but incorporates correlations which might be used in order to approximate the two variable covariance functions. While within the variational bound approximation, both approximations appear independent from each other we will, in the following develop an approach for combining two complimentary approximations which “communicate” by matching the corresponding expectations of the functions $\mathbf{g}(\mathbf{x})$.

2.1 Notation

Throughout the paper, densities $p(\mathbf{x})$ are assumed relative to the Lebesgue measure $d\mathbf{x}$ in R^N . Other choices, such as the counting measure, may lead to alternative approximations for discrete variables. We will denote the expectation of a function $h(\mathbf{x})$ with respect to a

2. This excludes many interesting models, for example mixture models, where tractability cannot be achieved with one split. These models can be treated by applying the approximation repeatedly. But for sake of clarity we will limit the treatment here to only one split.

density p by brackets

$$\langle h(\mathbf{x}) \rangle = \int d\mathbf{x} p(\mathbf{x}) h(\mathbf{x}) = \frac{1}{Z} \int d\mathbf{x} f(\mathbf{x}) h(\mathbf{x}) , \tag{8}$$

where, in cases of ambiguity, the density will appear as a subscript, like in $\langle h(\mathbf{x}) \rangle_p$. One of the strengths of our formalism is to allow for a treatment of discrete and continuous random variables within the same approach.

Example: Ising variables Discrete random variables can be described using Dirac distributions in the densities. For examples, the density of N independent Ising variables $x_i \in \{-1, +1\}$ which occur with equal probabilities (one-half) has the density

$$p(\mathbf{x}) = \prod_{i=1}^N \left[\frac{1}{2} \delta(x_i + 1) + \frac{1}{2} \delta(x_i - 1) \right] . \tag{9}$$

3. Expectation Consistent Free Energy Approximation

In this section we will derive an approximation for $-\ln Z$, the negative log-partition function also called the (Helmholtz) free energy. We will use an approximating distribution $q(\mathbf{x})$ of the type eq. (4) and split the exact free energy into a corresponding part $-\ln Z_q$ plus a rest which will be further approximated. The split is obtained by writing

$$\begin{aligned} Z &= Z_q \frac{Z}{Z_q} = Z_q \frac{\int d\mathbf{x} f_r(\mathbf{x}) f_q(\mathbf{x}) \exp((\boldsymbol{\lambda}_q - \boldsymbol{\lambda}_q)^T \mathbf{g}(\mathbf{x}))}{\int d\mathbf{x} f_q(\mathbf{x}) \exp \boldsymbol{\lambda}_q^T \mathbf{g}(\mathbf{x})} \\ &= Z_q \langle f_r(\mathbf{x}) \exp(-\boldsymbol{\lambda}_q^T \mathbf{g}(\mathbf{x})) \rangle_q , \end{aligned} \tag{10}$$

where

$$Z_q(\boldsymbol{\lambda}_q) = \int d\mathbf{x} f_q(\mathbf{x}) \exp(\boldsymbol{\lambda}_q^T \mathbf{g}(\mathbf{x})) . \tag{11}$$

This expression can be used to derive a variational bound to the free energy $-\ln Z$. Applying Jensen's inequality $\ln \langle f(\mathbf{x}) \rangle \geq \langle \ln f(\mathbf{x}) \rangle$ we arrive at

$$-\ln Z \leq -\ln Z^{\text{var}} = -\ln Z_q - \langle \ln f_r(\mathbf{x}) \rangle_q + \boldsymbol{\lambda}_q^T \langle \mathbf{g}(\mathbf{x}) \rangle_q . \tag{12}$$

The optimal value for $\boldsymbol{\lambda}_q$ is found by minimizing this upper bound.

Our new approximation is obtained by arguing that *one may do better by retaining the $f_r(\mathbf{x}) \exp(-\boldsymbol{\lambda}_q^T \mathbf{g}(\mathbf{x}))$ expression in eq. (10) but instead changing the distribution we use in averaging.* Hence, we replace the average with respect to $q(\mathbf{x})$ with an average using a distribution $s(\mathbf{x})$ containing the same exponential form

$$s(\mathbf{x}) = \frac{1}{Z_s(\boldsymbol{\lambda}_s)} \exp(\boldsymbol{\lambda}_s^T \mathbf{g}(\mathbf{x})) .$$

Given a sensible strategy for choosing the parameters $\boldsymbol{\lambda}_s$ and $\boldsymbol{\lambda}_q$, we expect that this approach in most cases gives a more precise approximation than the corresponding variational bound. Qualitatively, the more one can retain of the intractable function in the averaging

the closer the result will to the exact partition function. It is difficult to make this statement quantitative and general. However, the method gives nontrivial results for a variety of cases where the variational bound would be simply infinite! This always happens, when f_q is Gaussian and f_r vanishes on a set which has nonzero probability with respect to the density f_q . Examples are when f_r is discrete or contains likelihoods which vanish in certain regions as in noise-free Gaussian process classifiers (Oppel and Winther, 1999). Our approximation is further supported by the fact that for specific choices of f_r and f_q it is equivalent to the adaptive TAP (ADATAP) approximation (Oppel and Winther, 2001a,b). ADATAP (unlike the variational bound) gives exact results for certain statistical ensembles of distributions in an asymptotic (thermodynamic) limit studied in statistical physics.

Using s instead of q , we arrive at the approximation for $-\ln Z$ which depends upon two sets of parameters λ_q and λ_s :

$$\begin{aligned}
 -\ln Z^{\text{EC}}(\lambda_q, \lambda_s) &= -\ln Z_q - \ln \langle f_r(\mathbf{x}) \exp(-\lambda_q^T \mathbf{g}(\mathbf{x})) \rangle_s \\
 &= -\ln \int d\mathbf{x} f_q(\mathbf{x}) \exp(\lambda_q^T \mathbf{g}(\mathbf{x})) - \ln \int d\mathbf{x} f_r(\mathbf{x}) \exp((\lambda_s - \lambda_q)^T \mathbf{g}(\mathbf{x})) \\
 &\quad + \ln \int d\mathbf{x} \exp(\lambda_s^T \mathbf{g}(\mathbf{x})) .
 \end{aligned} \tag{13}$$

Here we have utilized our additional assumption, that also f_r is tractable with respect to the exponential family and thus $Z_r = \int d\mathbf{x} f_r(\mathbf{x}) \exp((\lambda_s - \lambda_q)^T \mathbf{g}(\mathbf{x}))$ can be computed in polynomial time. Eq. (13) leaves two sets of parameters λ_q and λ_s to be determined. We expect that eq. (13) is a sensible approximation as long as $s(\mathbf{x})$ shares some key properties with q , for which we choose the *matching of the moments* $\langle \mathbf{g}(\mathbf{x}) \rangle_q = \langle \mathbf{g}(\mathbf{x}) \rangle_s$. This will fix λ_s as a function of λ_q . Second, we know that the exact expression eq. (10) is independent of the value of λ_q . If the replacement of $q(x)$ by $s(x)$ yields a good approximation, one would still expect that eq. (13) is a fairly flat function of λ_q (after eliminating λ_s) in a certain region. Hence, it makes sense to require that an optimized approximation should make eq. (13) *stationary* with respect to variations of λ_q . This does not imply that we are expecting a local minimum of eq. (13), see also section 3.1, but saddle points could occur. Since we are not after a bound on the free energy, this is not necessarily a disadvantage of the method. Readers who feel uneasy with this argument, might find the alternative, dual derivation (using the Gibbs free energy) in appendix B more appealing.

Both conditions can be summarized by the *expectation consistency* (EC) conditions

$$\frac{\partial \ln Z^{\text{EC}}}{\partial \lambda_q} = 0 \quad : \quad \langle \mathbf{g}(\mathbf{x}) \rangle_q = \langle \mathbf{g}(\mathbf{x}) \rangle_r \tag{14}$$

$$\frac{\partial \ln Z^{\text{EC}}}{\partial \lambda_s} = 0 \quad : \quad \langle \mathbf{g}(\mathbf{x}) \rangle_r = \langle \mathbf{g}(\mathbf{x}) \rangle_s \tag{15}$$

for the three approximating distributions

$$q(\mathbf{x}) = \frac{1}{Z_q(\lambda_q)} f_q(\mathbf{x}) \exp(\lambda_q^T \mathbf{g}(\mathbf{x})) \tag{16}$$

$$r(\mathbf{x}) = \frac{1}{Z_r(\lambda_r)} f_r(\mathbf{x}) \exp(\lambda_r^T \mathbf{g}(\mathbf{x})) \quad \text{with} \quad \lambda_r = \lambda_s - \lambda_q \tag{17}$$

$$s(\mathbf{x}) = \frac{1}{Z_s(\lambda_s)} \exp(\lambda_s^T \mathbf{g}(\mathbf{x})) . \tag{18}$$

The corresponding EC approximation of the free energy is then

$$-\ln Z \approx -\ln Z^{\text{EC}} = -\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_s - \boldsymbol{\lambda}_q) + \ln Z_s(\boldsymbol{\lambda}_s) \quad (19)$$

where $\boldsymbol{\lambda}_q$ and $\boldsymbol{\lambda}_s$ are chosen such that the partial derivatives of the right hand side vanish.

3.1 Properties of the EC approximation

Invariances Although our derivation started with approximating one of the two factors f_q and f_r by an exponential, the final approximation is completely symmetric in the factors f_q and f_r . We could have chosen to define q in terms of f_r and still got the same final result. If f contains multiplicative terms which are of the form $\exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}))$ for some fixed $\boldsymbol{\lambda}$, we are free to include them either in f_q or f_r without changing the approximation. This can be easily shown by redefining $\boldsymbol{\lambda}_q \rightarrow \boldsymbol{\lambda}_q \pm \boldsymbol{\lambda}$.

Derivatives with respect to parameters. The following is a useful result about the derivative of $-\ln Z^{\text{EC}}$ with respect to a parameter t in the density $p(\mathbf{x})$. Setting $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_s)$, we get

$$\frac{d \ln Z^{\text{EC}}(t)}{dt} = \frac{\partial \ln Z^{\text{EC}}(\boldsymbol{\lambda}, t)}{\partial t} + \left(\frac{\partial \ln Z^{\text{EC}}(\boldsymbol{\lambda}, t)}{\partial \boldsymbol{\lambda}} \right) \frac{d\boldsymbol{\lambda}^T}{dt} = \frac{\partial \ln Z^{\text{EC}}(\boldsymbol{\lambda}, t)}{\partial t}, \quad (20)$$

where the second equality holds at the stationary point. The important message is that we only need to take the explicit t dependence into account, i.e. we can keep the stationary values $\boldsymbol{\lambda}$ fixed upon differentiation. This property can also be useful when optimizing the free energy with respect to parameters in the empirical Bayes framework.

Relation to the variational bound. Applying Jensen's inequality to (13) yields

$$\begin{aligned} -\ln Z^{\text{EC}}(\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_s) &= -\ln Z_q - \ln \langle f_r(\mathbf{x}) \exp(-\boldsymbol{\lambda}_q^T \mathbf{g}(\mathbf{x})) \rangle_s \\ &\geq -\ln Z_q - \langle \ln f_r(\mathbf{x}) \rangle_s + \boldsymbol{\lambda}_q^T \langle \mathbf{g}(\mathbf{x}) \rangle_s. \end{aligned}$$

Hence, if f_r and $\mathbf{g}(\mathbf{x})$ are defined in such a way that the matching of the moments $\langle \mathbf{g}(\mathbf{x}) \rangle_s = \langle \mathbf{g}(\mathbf{x}) \rangle_q$ implies $\langle \ln f_r(\mathbf{x}) \rangle_q = \langle \ln f_r(\mathbf{x}) \rangle_s$ then the rhs of the inequality is equal to the variational (bound) free energy eq. (12) for fixed $\boldsymbol{\lambda}_q$. This will be the case for the models discussed in this paper. Of course, this does not imply any relation between $-\ln Z^{\text{EC}}$ and the true free energy. The similarity of EC to the variational bound approximation should also be interpreted with care. One could be tempted to try solving the EC stationarity conditions by eliminating $\boldsymbol{\lambda}_s$, i.e. enforcing the moment constraints between q and s , and minimizing the free energy approximation $-\ln Z^{\text{EC}}(\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_s(\boldsymbol{\lambda}_q))$ with respect to $\boldsymbol{\lambda}_q$, as in the variational bound method. Simple counter examples show however that this function may be unbounded from below and that the stationary point may not even be a local minimum.

Non-convexity. The log-partition functions $\ln Z_{q,r,s}(\boldsymbol{\lambda})$ are the *cumulant generating functions* of the random variables $\mathbf{g}(\mathbf{x})$. Hence, they are differentiable and convex functions on their domains of definition, i.e.

$$\mathbf{H} = \frac{\partial^2 \ln Z}{\partial \boldsymbol{\lambda}^T \partial \boldsymbol{\lambda}} = \langle \mathbf{g}(\mathbf{x}) \mathbf{g}(\mathbf{x})^T \rangle - \langle \mathbf{g}(\mathbf{x}) \rangle \langle \mathbf{g}(\mathbf{x}) \rangle^T$$

is positive semi-definite. It follows for fixed λ_s that eq. (19) is concave in the variable λ_q , and there is only a single solution to eq. (14) corresponding to a maximum of $-\ln Z_q(\lambda_q) - \ln Z_r(\lambda_s - \lambda_q)$. On the other hand, eq. (19) is a sum of a concave and a convex function of λ_s . Thus, unfortunately there may be more than one stationary point, a property which the EC approach shares with other approximations such as the variational Bayes and the Bethe–Kikuchi methods. Nevertheless, we can use a double loop algorithm which alternates between solving the concave maximization problem for λ_q at fixed λ_s and updating λ_s given the values of the moments $\langle \mathbf{g}(\mathbf{x}) \rangle_r = \langle \mathbf{g}(\mathbf{x}) \rangle_q$ at fixed λ_q . We will show in Section 5 and in Appendix B that such a simple heuristic leads to convergence to a stationary point assuming that a certain cost function is bounded from below.

4. EC Free Energies – Examples

In this section we derive the EC free energy for a specific model, the quadratic, and discuss several possible choices for the consistent statistics $\langle \mathbf{g}(\mathbf{x}) \rangle$.

4.1 Tractable Free Energies

Our approach applies most naturally to a class of models for which the distribution of random variables \mathbf{x} can be written as a product of a factorizing part eq. (6) and “Gaussian part” eq. (7).³ The choice of $\mathbf{g}(\mathbf{x})$ is then guided by the need to make the computation of the EC free energy, eq. (19), tractable. The “Gaussian part” stays tractable as long as we take $\langle \mathbf{g}(\mathbf{x}) \rangle$ to contain first and second moments of \mathbf{x} . It will usually be a good idea to take all first moments, but we have a freedom in choosing the amount of consistency and the number of second order moments in $\langle \mathbf{g}(\mathbf{x}) \rangle$. To keep Z_q tractable (assuming f_q it is not Gaussian), a restriction to diagonal moments, i.e. $\langle x_i^2 \rangle$ will be sufficient. When variables are discrete, it is also possible to include second moments $\langle x_i x_j \rangle$ for pairs of variables located at the edges \mathcal{G} of a tree.

The following three choices represent approximations of increasing complexity:

- Diagonal restricted: consistency on $\langle x_i \rangle$, $i = 1, \dots, N$ and $\sum_i \langle x_i^2 \rangle$.

$$\mathbf{g}(\mathbf{x}) = \left(x_1, \dots, x_N, -\sum_i \frac{x_i^2}{2} \right) \quad \text{and} \quad \boldsymbol{\lambda} = (\gamma_1, \dots, \gamma_N, \Lambda)$$

- Diagonal: consistency on $\langle x_i \rangle$ and $\langle x_i^2 \rangle$, $i = 1, \dots, N$

$$\mathbf{g}(\mathbf{x}) = \left(x_1, -\frac{x_1^2}{2}, \dots, x_N, -\frac{x_N^2}{2} \right) \quad \text{and} \quad \boldsymbol{\lambda} = (\gamma_1, \Lambda_1, \dots, \gamma_N, \Lambda_N)$$

- Spanning tree: as above and additional consistency of correlations $\langle x_i x_j \rangle$ defined on a spanning tree $(ij) \in \mathcal{G}$. Since we are free to move the terms $J_{ij} x_i x_j$ with $(ij) \in \mathcal{G}$ from the Gaussian term f_r into the term f_q , without changing the approximation, we find that the number of interaction terms which have to be approximated using the

3. A generalization where f_q factorizes into tractable “potentials” ψ_α defined on disjoint subsets \mathbf{x}_α of \mathbf{x} is also straightforward.

complementary Gaussian density is reduced. If the tree is chosen in such a way as to include the most important couplings (defined in a proper fashion), one can expect that the approximation will be improved significantly.

It is of course also possible to go beyond a spanning tree to treat a larger part of the marginalization exactly. We will next give explicit expressions for some free energies which will be used later for the EC approximation.

Independent Ising random variables. Here, we consider N independent Ising variables $x_i \in \{-1, +1\}$:

$$f(\mathbf{x}) = \prod_{i=1}^N \psi_i(x_i) \quad \text{with} \quad \psi_i(x_i) = [\delta(x_i + 1) + \delta(x_i - 1)] . \quad (21)$$

For the case of diagonal moments we get $Z(\boldsymbol{\lambda}) = \prod_i Z_i(\boldsymbol{\lambda}_i)$, $\boldsymbol{\lambda}_i = (\gamma_i, \Lambda_i)$:

$$Z_i(\boldsymbol{\lambda}_i) = \int dx_i \psi_i(x_i) e^{\gamma_i x_i - \Lambda_i x_i^2/2} = 2 \cosh(\gamma_i) e^{-\Lambda_i/2} . \quad (22)$$

Multivariate Gaussian. Consider a Gaussian model: $p(\mathbf{x}) = \frac{1}{Z} e^{\mathbf{x}^T \mathbf{J} \mathbf{x} + \boldsymbol{\theta}^T \mathbf{x}}$. We introduce an arbitrary set of first moments $\langle x_i \rangle$ and second moments $-\langle x_i x_j \rangle/2$ with conjugate variables $\boldsymbol{\gamma}$ and $\boldsymbol{\Lambda}$. Here it is understood, that entries of $\boldsymbol{\gamma}$ and $\boldsymbol{\Lambda}$ corresponding to the non-fixed moments are set equal to zero. $\boldsymbol{\Lambda}$ is chosen to be a symmetric matrix, $\Lambda_{ij} = \Lambda_{ji}$, for notational convenience. The resulting free energy is

$$\ln Z(\boldsymbol{\gamma}, \boldsymbol{\Lambda}) = \frac{N}{2} \ln 2\pi - \frac{1}{2} \ln \det(\boldsymbol{\Lambda} - \mathbf{J}) + \frac{1}{2} (\boldsymbol{\gamma} + \boldsymbol{\theta})^T (\boldsymbol{\Lambda} - \mathbf{J})^{-1} (\boldsymbol{\gamma} + \boldsymbol{\theta}) .$$

The free energies for binary and Gaussian tree graphs are given in Appendix C.

4.2 EC Approximation

We can now write down the explicit expression for the free energy, eq. (19) for the model eqs. (6) and (7) with diagonal moments using the result for the Gaussian model:

$$\begin{aligned} -\ln Z^{\text{EC}} &= -\sum_i \ln \int dx_i \psi_i(x_i) e^{\gamma_{q,i} x_i - \Lambda_{q,i} x_i^2/2} + \frac{1}{2} \ln \det(\boldsymbol{\Lambda}_s - \boldsymbol{\Lambda}_q - \mathbf{J}) \\ &\quad - \frac{1}{2} (\boldsymbol{\theta} + \boldsymbol{\gamma}_s - \boldsymbol{\gamma}_q)^T (\boldsymbol{\Lambda}_s - \boldsymbol{\Lambda}_q - \mathbf{J})^{-1} (\boldsymbol{\theta} + \boldsymbol{\gamma}_s - \boldsymbol{\gamma}_q) - \frac{1}{2} \sum_i \left(\ln \Lambda_{s,i} - \frac{\gamma_{s,i}^2}{\Lambda_{s,i}} \right) \end{aligned} \quad (23)$$

where $\boldsymbol{\lambda}_q$ and $\boldsymbol{\lambda}_s$ are chosen to make $-\ln Z^{\text{EC}}$ stationary. The $\ln Z_s(\boldsymbol{\lambda}_s)$ term is obtained from the general Gaussian model setting $\boldsymbol{\theta} = \mathbf{0}$ and $\mathbf{J} = \mathbf{0}$.

Generating moments. Derivatives of the free energy with respect to parameters provide a simple way for generating expectations of functions of the random variable \mathbf{x} . We will explain the method for the second moments $\langle x_i x_j \rangle$ of the model defined by the factorization eqs. (6) and (7). If we consider $p(\mathbf{x})$ as a function of the parameter J_{ij} , we get after a short calculation

$$\frac{d \ln Z(\boldsymbol{\lambda}, J_{ij})}{d J_{ij}} = \frac{1}{2} \langle x_i x_j \rangle . \quad (24)$$

Here we assume that the coupling matrix \mathbf{J} is augmented to a full matrix with the auxiliary elements set to zero at the end. Evaluating the left hand side of eq. (24) within the EC approximation eq. (23) and using eq. (20) yields

$$\langle \mathbf{x}\mathbf{x}^T \rangle - \langle \mathbf{x} \rangle \langle \mathbf{x} \rangle^T = (\mathbf{\Lambda}_s - \mathbf{\Lambda}_q - \mathbf{J})^{-1} . \quad (25)$$

The result eq. (25) could have also obtained by computing the covariance matrix directly from the Gaussian approximating density $r(\mathbf{x})$. We have consistency between $r(\mathbf{x})$ and $q(\mathbf{x})$ on the second order moments included in $\mathbf{g}(\mathbf{x})$, but for those not included, one can argue on quite general grounds that $r(\mathbf{x})$ will be more precise than $q(\mathbf{x})$ (Opper and Winther, 2004). Similarly, one may hope that higher order diagonal moments or even the entire marginal density of variables can be well approximated using the density $q(\mathbf{x})$. An application which shows the quality of this approximation can be found in Malzahn and Opper (2003).

5. Algorithms

This section deals with the task of solving the EC optimization problem, that is solving the consistency conditions eqs. (14) and (15): $\langle \mathbf{g}(\mathbf{x}) \rangle_q = \langle \mathbf{g}(\mathbf{x}) \rangle_r = \langle \mathbf{g}(\mathbf{x}) \rangle_s$ for the three distributions q , r and s , eqs. (16)-(18). As already discussed in section 3, the EC free energy is not a concave function in the parameters λ_q , λ_s and one may have to resort to double loop approaches (Welling and Teh, 2003; Yuille, 2002; Heskes et al., 2003; Yuille and Rangarajan, 2003). Heskes and Zoeter (2002) were the first to apply double loop algorithms EC type of approximations. Since the double loop approaches may be slow in practice it is also of interest to define single loop algorithms that come with no warranty, but in many practical cases will converge fast. A pragmatic strategy is thus to first try a single loop algorithm and switch to a double loop when necessary. In the following we first discuss the algorithms in general and then specialize to the model eqs. (6) and (7).

5.1 Single Loop Algorithms

The single loop approaches typically are of the form of propagation algorithms which send “messages” back and forth between the two distributions $q(\mathbf{x})$ and $r(\mathbf{x})$. In each step the “separator” or “overlap distribution” $s(\mathbf{x})$ ⁴ is updated to be consistent with either q or r depending upon which way we are propagating. This corresponds to an Expectation Propagation style scheme with two terms, see also Appendix D. Iteration t of the algorithm can be sketched as follows:

1. Send message from r to q
 - Calculate separator $s(\mathbf{x})$: Solve for λ_s : $\langle \mathbf{g}(\mathbf{x}) \rangle_s = \boldsymbol{\mu}_r(t-1) \equiv \langle \mathbf{g}(\mathbf{x}) \rangle_{r(t-1)}$
 - Update $q(\mathbf{x})$: $\lambda_q(t) := \lambda_s - \lambda_r(t-1)$
2. Send message from q to r
 - Calculate separator $s(\mathbf{x})$: Solve for λ_s : $\langle \mathbf{g}(\mathbf{x}) \rangle_s = \boldsymbol{\mu}_q(t) \equiv \langle \mathbf{g}(\mathbf{x}) \rangle_{q(t)}$

4. These names are chosen because $s(\mathbf{x})$ plays the same role as the separator potential in the junction tree algorithm and as the overlap distribution in the Bethe approximation.

- Update $r(\mathbf{x})$: $\lambda_r(t) := \lambda_s - \lambda_q(t)$.

Here $r(t)$ and $q(t)$ denote the distributions q and r computed with the parameters $\lambda_r(t)$ and $\lambda_q(t)$. Convergence is reached when $\mu_r = \mu_q$ since each parameter update ensures $\lambda_r = \lambda_s - \lambda_q$. Several modifications of the above algorithm are possible. First of all a “damping factor” (or “learning rate”) η can be introduced on both or one of the parameter updates. Secondly we can abandon the parallel update and solve sequentially for factors containing only subsets of parameters.

5.2 Single Loop Algorithms for Quadratic Model

In the following we will explain details of the algorithm for the quadratic model eqs. (6) and (7) with consistency for first and second diagonal moments, corresponding to the EC free energy eq. (23). We will also briefly sketch the algorithm for moment consistency on a spanning tree. In appendix D we give the algorithmic recipes for a sequential algorithm for the factorized approximation and a parallel algorithm for tree approximation. These are simple, fast and quite reliable.

For the diagonal choice of $\mathbf{g}(\mathbf{x})$, $s(\mathbf{x})$ is simply the product of univariate Gaussians: $s(\mathbf{x}) = \prod_i s_i(x_i)$ and $s_i(x_i) \propto \exp(\gamma_{s,i}x_i - \Lambda_{s,i}x_i^2/2)$. Solving for $s(\mathbf{x})$ in terms of the moments of q and r , respectively, corresponds to a simple marginal moment matching to the univariate Gaussian $\propto \exp(-(x_i - m_i)^2/2v_i)$: $\gamma_{s,i} := m_i/v_i$ and $\Lambda_{s,i} := 1/v_i$. $r(\mathbf{x})$ is a multivariate Gaussian with covariance, eq. (25), $\chi_r \equiv (\mathbf{\Lambda}_r - \mathbf{J})^{-1}$ and mean $\mathbf{m}_r = \chi_r \gamma_r$. Matching the moments with $r(\mathbf{x})$ gives $m_i := m_{r,i}$ and $v_i := \chi_{r,ii}$. The most expensive operation of the algorithm is the calculation of the moments of $r(\mathbf{x})$ which is $\mathcal{O}(N^3)$ because $\chi_r = (\mathbf{\Lambda}_r - \mathbf{J})^{-1}$ has to be recalculated after each update of λ_r . $q(\mathbf{x})$ is a factorized non-Gaussian distribution for which we have to obtain the mean and variance and match as above.

The spanning tree algorithm is only slightly more complicated. Now $s(\mathbf{x})$ is a Gaussian distribution on a spanning tree. Solving for λ_s can be performed in linear complexity in N using the tree decomposition of the free energy, see appendix C. $r(\mathbf{x})$ is still a full multivariate Gaussian and inferring the moments of the spanning tree distribution $q(\mathbf{x})$ is $\mathcal{O}(N)$ using message passing (MacKay, 2003).

5.3 Double Loop Algorithm

Since the EC free energy $-\ln Z^{\text{EC}}(\lambda_q, \lambda_s)$ is concave in λ_q , we can attempt a solution of the stationarity problem eqs. (14) and (15), by first solving the *concave maximization* problem

$$F(\lambda_s) \equiv \max_{\lambda_q} \{-\ln Z^{\text{EC}}(\lambda_q, \lambda_s)\} = \max_{\lambda_q} \{-\ln Z_q(\lambda_q) - \ln Z_r(\lambda_s - \lambda_q)\} + \ln Z_s(\lambda_s) \quad (26)$$

and subsequently finding a solution to the equation

$$\frac{\partial F(\lambda_s)}{\partial \lambda_s} = 0. \quad (27)$$

Since $F(\lambda_s)$ is in general neither a convex nor a concave function, there might be many solutions to this equation.

The double loop algorithm aims at finding a solution iteratively. It starts with an arbitrary admissible value $\boldsymbol{\lambda}_s(0)$ and iterates two elementary procedures for updating $\boldsymbol{\lambda}_s$ and $\boldsymbol{\lambda}_q$ aiming at matching the moments between the distribution q, r and s . Assume that at iteration step t $\boldsymbol{\lambda}_s = \boldsymbol{\lambda}_s(t)$, then iterate over the two steps

1. **Solve the concave maximization problem eq. (26)** yielding the update

$$\boldsymbol{\lambda}_q(t) = \operatorname{argmax}_{\boldsymbol{\lambda}_q} \left\{ -\ln Z^{\text{EC}}(\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_s(t)) \right\} . \quad (28)$$

With this update, we achieve equality of the moments

$$\boldsymbol{\mu}(t) \equiv \langle \mathbf{g}(\mathbf{x}) \rangle_{q(t)} = \langle \mathbf{g}(\mathbf{x}) \rangle_{r(t)} . \quad (29)$$

2. **Update $\boldsymbol{\lambda}_s$ as**

$$\boldsymbol{\lambda}_s(t+1) = \operatorname{argmin}_{\boldsymbol{\lambda}_s} \left\{ -\boldsymbol{\lambda}_s^T \boldsymbol{\mu}(t) + \ln Z_s(\boldsymbol{\lambda}_s) \right\} \quad (30)$$

which is a convex minimization problem. This yields $\langle \mathbf{g}(\mathbf{x}) \rangle_{s(t+1)} = \boldsymbol{\mu}(t)$.

To discuss convergence of these iterations, we prove that $F(\boldsymbol{\lambda}_s(t))$ for $t = 0, 1, 2, \dots$ is a nondecreasing sequence:

$$\begin{aligned} F(\boldsymbol{\lambda}_s(t)) &= \max_{\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_r} \left\{ -\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_r) + \ln Z_s(\boldsymbol{\lambda}_s) + (\boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r - \boldsymbol{\lambda}_s(t))^T \boldsymbol{\mu}(t) \right\} \quad (31) \\ &\geq \max_{\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_r} \left\{ -\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_r) + (\boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r)^T \boldsymbol{\mu}(t) + \min_{\boldsymbol{\lambda}_s} \left(-\boldsymbol{\lambda}_s^T \boldsymbol{\mu}(t) + \ln Z_s(\boldsymbol{\lambda}_s) \right) \right\} \\ &= \max_{\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_r} \left\{ -\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_r) + \ln Z_s(\boldsymbol{\lambda}_s(t+1)) + (\boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r - \boldsymbol{\lambda}_s(t+1))^T \boldsymbol{\mu}(t) \right\} \\ &\geq \max_{\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_r | \boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r = \boldsymbol{\lambda}_s(t+1)} \left\{ -\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_r) \right\} + \ln Z_s(\boldsymbol{\lambda}_s(t+1)) \\ &= F(\boldsymbol{\lambda}_s(t+1)) . \end{aligned}$$

The first equality follows from the fact that $\boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r - \boldsymbol{\lambda}_s(t) = 0$ and that at the maximum we have matching moments $\boldsymbol{\mu}(t)$ for the q and r distributions. The next inequality is true because we do not increase $-\boldsymbol{\lambda}_s^T \boldsymbol{\mu}(t) + \ln Z_s(\boldsymbol{\lambda}_s)$ by minimizing. The next equality implements the definition of eq. (30). The final inequality follows because we maximize over a restricted set. Hence, when F is bounded from below we will get convergence.

Hence, the double loop algorithm attempts in fact a minimization of $F(\boldsymbol{\lambda}_s)$. It is not clear a priori why we should search for a minimum rather than a maximum or any other critical value. However, a reformulation of the EC approach given in Appendix B shows that we can interpret $F(\boldsymbol{\lambda}_s)$ as an upper bound on an approximation to the so-called Gibbs free energy which is the Lagrange dual to the Helmholtz free energy from which the desired moments are derived by minimization.

5.4 Double Loop Algorithms for the Quadratic Model

The outer loop optimization problem (step 2 above) for $\boldsymbol{\lambda}_s$ is identical to the one for the single loop algorithm. The concave optimization problem of the inner loop for $\mathcal{L}(\boldsymbol{\lambda}_q) \equiv$

$-\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_s(t) - \boldsymbol{\lambda}_q)$ (step 1 above) can be solved by standard techniques from convex optimization (Vandenberghe et al., 1998; Boyd and Vandenberghe, 2004). Here we will describe a sequential approach that exploits the fact that updating only one element in $\boldsymbol{\Lambda}_r = \boldsymbol{\Lambda}_s(t) - \boldsymbol{\Lambda}_q$ (or in spanning tree case a two-by-two sub-matrix) is a rank one (or rank two) update of $\boldsymbol{\chi}_r = (\boldsymbol{\Lambda}_r - \mathbf{J})^{-1}$ that can be performed in $\mathcal{O}(N^2)$.

Specializing to the quadratic model with diagonal $\mathbf{g}(\mathbf{x})$ we have to maximize

$$\begin{aligned}
 \mathcal{L}(\boldsymbol{\lambda}_q) &= - \sum_i \ln \int dx_i \psi_i(x_i) \exp \left[\gamma_{q,i} x_i - \frac{1}{2} \Lambda_{q,i} x_i^2 \right] \\
 &\quad - \ln \int d\mathbf{x} \exp \left[-\frac{1}{2} \mathbf{x}^T (\boldsymbol{\Lambda}_s(t) - \boldsymbol{\Lambda}_q - \mathbf{J}) \mathbf{x} + (\boldsymbol{\gamma}_s(t) - \boldsymbol{\gamma}_q)^T \mathbf{x} \right]
 \end{aligned}$$

with respect to γ_q and Λ_q . We aim at a sequential approach where we optimize the variables for one element in \mathbf{x} , say the i th. We can isolate $\gamma_{q,i}$ and $\Lambda_{q,i}$ in the Gaussian term to obtain a reduced optimization problem:

$$\begin{aligned}
 \mathcal{L}(\gamma_{q,i}, \Lambda_{q,i}) &= \text{const} + \frac{1}{2} \ln [1 - v_{r,i} (\Lambda_{q,i}^0 - \Lambda_{q,i})] - \frac{(\gamma_{q,i}^0 - \gamma_{q,i} - m_{r,i}/v_{r,i})^2}{2(1/v_{r,i} + \Lambda_{q,i}^0 - \Lambda_{q,i})} \\
 &\quad - \log \int dx_i \psi_i(x_i) \exp \left[\gamma_{q,i} x_i + \frac{1}{2} \Lambda_{q,i} x_i^2 \right], \tag{32}
 \end{aligned}$$

where superscript 0 denotes current values of the parameters and we have set $m_{r,i} = \langle x_i \rangle_r = [(\boldsymbol{\Lambda}_r^0 - \mathbf{J})^{-1} \boldsymbol{\gamma}_r^0]_i$ and $v_{r,i} = \langle x_i^2 \rangle_r - m_{r,i}^2 = [(\boldsymbol{\Lambda}_{r,i}^0 - \mathbf{J})^{-1}]_{ii}$, with $\boldsymbol{\lambda}_r = \boldsymbol{\lambda}_s(t) - \boldsymbol{\lambda}_q$. Introducing the corresponding two first moments for $q_i(x_i)$

$$m_{q,i} = m_{q,i}(\gamma_{q,i}, \Lambda_{q,i}) = \langle x_i \rangle_q = \frac{1}{Z_{q_i}} \int dx_i x_i \psi_i(x_i) \exp \left[\gamma_{q,i} x_i - \frac{1}{2} \Lambda_{q,i} x_i^2 \right] \tag{33}$$

$$v_{q,i} = v_{q,i}(\gamma_{q,i}, \Lambda_{q,i}) = \langle x_i^2 \rangle_q - m_{q,i}^2 \tag{34}$$

we can write the stationarity condition for $\gamma_{q,i}$ and $\Lambda_{q,i}$ as:

$$\gamma_{q,i} + \frac{m_{q,i}}{v_{q,i}} = \gamma_{q,i}^0 + \frac{m_{r,i}}{v_{r,i}} \tag{35}$$

$$\Lambda_{q,i} + \frac{1}{v_{q,i}} = \Lambda_{q,i}^0 + \frac{1}{v_{r,i}} \tag{36}$$

collecting variable terms and constant terms on the lhs and rhs, respectively. These two equations can be solved very fast with a Newton method. For binary variables the equations decouple since $m_{q,i} = \tanh(\gamma_{q,i})$ and $v_{q,i} = 1 - m_{q,i}^2$ and we are left with a one dimensional problem.

Typically, solving these two non-linear equations are not the most computationally expensive steps because after these have been solved, the first two moments of the r -distribution have to be recalculated. This final step can be performed using the matrix inversion lemma (or Sherman-Morrison formula) to reduce the computation to $\mathcal{O}(N^2)$. The matrix of second moments $\boldsymbol{\chi}_r = (\boldsymbol{\Lambda}_r - \mathbf{J})^{-1}$ is thus updated as:

$$\boldsymbol{\chi}_r := \boldsymbol{\chi}_r - \frac{\Delta \Lambda_{r,i}}{1 + \Delta \Lambda_{r,i} [\boldsymbol{\chi}_r]_{ii}} [\boldsymbol{\chi}_r]_i [\boldsymbol{\chi}_r]_i^T, \tag{37}$$

where $\Delta\Lambda_{r,i} = -\Delta\Lambda_{q,i} = -(\Lambda_{q,i} - \Lambda_{q,i}^0) = \frac{1}{v_{q,i}} - \frac{1}{v_{r,i}}$ and $[\chi_r]_i$ is defined to be the i th row in χ_r .

Note that the solution for $\Lambda_{q,i}$ is a coordinate ascent solution which has the nice property that if we initialize $\Lambda_{q,i}$ with an admissible value, i.e. with χ_r positive semi-definite then with this update χ_r will stay positive definite since the objective has an infinite barrier at $\det \chi_r = 0$.

6. Simulations

In this section we apply expectation consistent inference (EC) to the model of pair-wise connected Ising variables introduced in Section 4. We consider two versions of EC: “factorized” with $\mathbf{g}(\mathbf{x})$ containing all first and only diagonal second moments and the structured “spanning tree” version. The tree is chosen as a maximum spanning tree, where the maximum is defined over $|J_{ij}|$, i.e. choose as next pair of nodes to link, the (so far unlinked) pair with strongest absolute coupling $|J_{ij}|$ that will not cause a loop in the graph. The free energy is optimized with the parallel single loop algorithm described in section 5 and appendix D. Whenever non-convergence is encountered we switch to the double loop algorithm. We compare the performance of the two EC approximations with two other approaches for two different set-ups that have previously been used as benchmarks in the literature⁵.

In the first set of simulations we compare with the Bethe and Kikuchi approaches (Heskes et al., 2003). We consider $N = 10$ and choose constant “external fields” (observations) $\theta_i = \theta = 0.1$. The “couplings” J_{ij} are fully connected and generated independently at random according to $J_{ij} = \beta w_{ij} / \sqrt{N}$, the w_{ij} s are Gaussian with zero mean and unit variance. We consider eight different scalings $\beta = [0.10, 0.25, 0.50, 0.75, 1.00, 1.50, 2.00, 10.00]$. and compare one-variable marginals $p(x_i) = \frac{1+x_i m_i}{2}$ and the two-variable marginals $p(x_i, x_j) = \frac{x_i x_j C_{ij}}{4} + p(x_i)p(x_j)$ where C_{ij} is the covariance $C_{ij} = \langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle$. For EC, C_{ij} is given by eq. (25). In figure 1 we plot maximum absolute deviation (MAD) of our results from the exact marginals for different scaling parameters:

$$\begin{aligned} \text{MAD1} &= \max_i |p(x_i = 1) - p(x_i = 1|\text{Method})| \\ \text{MAD2} &= \max_{i,j} \max_{x_i=\pm 1, x_j=\pm 1} |p(x_i, x_j) - p(x_i, x_j|\text{Method})| . \end{aligned}$$

In figure 2 we compare estimates of the free energy. The results show that the simple factorized EC approach gives performance similar to (and in many case better than) the structured Bethe and Kikuchi approximations. The EC tree version is almost always better than the other approximations. The Kikuchi approximation is not uniquely defined, but depends upon the choice of “cluster-structure”. Different types of structures can give rise to quite different performance (Minka and Qi, 2004). The results given above is thus just to be taken as one realization of the Kikuchi method where the clusters are taken as all variable triplets. We expect the Kikuchi approximation to yield better results (probably better than EC in some cases) for an appropriate choice of sub-graphs, for example triangles forming a star for fully connected models and all squares for grids (Yedidia et al., 2001; Minka and Qi, 2004). EC can also be improved beyond trees as discussed in the Conclusion.

5. All results and programs are available from the authors.

The second test is the set-up proposed by Wainwright and Jordan (2003, 2005). The $N = 16$ nodes are either fully connected or connected to nearest neighbors in a 4-by-4 grid. The external field (observation) strengths θ_i are drawn from a *uniform* distribution $\theta_i \sim \mathcal{U}[-d_{\text{obs}}, d_{\text{obs}}]$ with $d_{\text{obs}} = 0.25$. Three types of coupling strength statistics are considered: repulsive (anti-ferromagnetic) $J_{ij} \sim \mathcal{U}[-2d_{\text{coup}}, 0]$, mixed $J_{ij} \sim \mathcal{U}[-d_{\text{coup}}, +d_{\text{coup}}]$ and attractive (ferromagnetic) $J_{ij} \sim \mathcal{U}[0, +2d_{\text{coup}}]$ with $d_{\text{coup}} > 0$. We compute the average absolute deviation on the marginals:

$$\text{AAD} = \frac{1}{N} \sum_i |p(x_i = 1) - p(x_i = 1|\text{method})|$$

over 100 trials testing the following methods: SP = sum-product (aka loopy belief propagation (BP) or Bethe approximation) and LD = log-determinant maximization (Wainwright and Jordan, 2003, 2005), EC factorized and EC tree. Results for SP and LD are taken from Wainwright and Jordan (2003). Note that instances where SP failed to converge were excluded from the results. A fact that is likely to bias the results in favor of SP. The results are summarized in table 6. The Bethe approximation always gives inferior results compared to EC. This might be a bit surprising for the sparsely connected grids. LD is a robust method which however seems to be limited in its achievable precision. EC tree is uniformly superior to all other approaches. It would be interesting to compare to the Kikuchi approximation which is known to give good results on grids.

A few comments about complexity, speed and rates of convergence: Both EC algorithms are $\mathcal{O}(N^3)$. For the $N = 16$ simulations typical wall clock times were 0.5 sec. for exact computation, half of that for the single-loop tree and one-tenth for the factorized single-loop. Convergence is defined to be when $|\langle \mathbf{g}(\mathbf{x}) \rangle_q - \langle \mathbf{g}(\mathbf{x}) \rangle_r|^2$ is below 10^{-12} . Double loop algorithms typically were somewhat slower (1-2 sec.) because a lot of outer loop iterations were required. This indicates that the bound optimized in the inner loop is very conservative for these binary problems. For the easy problems (small d_{coup}) all approaches converged. For the harder problems the factorized EP-style algorithms typically converged in 80-90 % of the cases. A greedy single-loop variant of the sequential double-loop algorithm, where the outer loop update is performed after every inner loop update, converged more often without being much slower than the EP-style algorithm. We treated the grid as a fully connected system yielding a complexity of $\mathcal{O}(N^3)$. Exploiting the structure using message passing, one can reduce the complexity of inference, i.e. calculating the covariance on the links, to $\mathcal{O}(N^2)$.

7. Conclusion and Outlook

We have introduced a novel method for approximate inference which tries to overcome limitations of previous approximations in dealing with the correlations of random variables. While we have demonstrated its accuracy in this paper only for a model with binary elements, it can also be applied to models with continuous random variables or hybrid models with both discrete and continuous variables (i.e. cases where further approximations are needed in order to apply Bethe/Kikuchi approaches).

We expect that our method becomes most powerful when certain tractable substructures of variables with strong dependencies can be identified in a model. Our approach would then

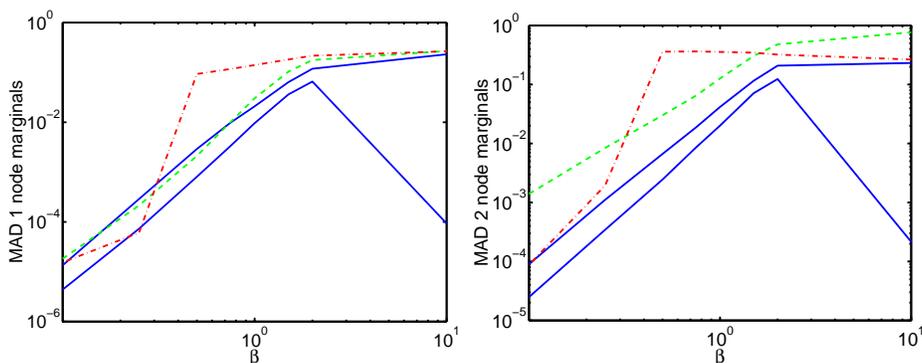


Figure 1: Maximal absolute deviation (MAD) for one- (left) and two-variable (right) marginals. EC factorized: upper full line (blue), EC tree: lower full line (blue), Bethe: dashed line (green) and Kikuchi: dash-dotted line (red).

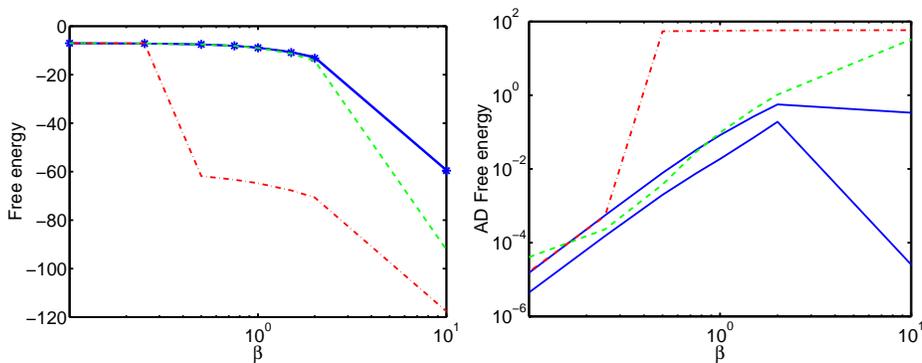


Figure 2: Left plot: free energy exact: stars, EC factorized and tree: full lines virtually on top on each others (blue), Bethe: dashed line (green) and Kikuchi: dash-dotted (red). Right: Absolute deviation (AD) for the three approximations, same line type (and color) as above. Lower full line is for the EC tree approximation.

allow us to deal well with the weaker dependencies between substructures. Better heuristics for determining the choice of substructures will also be useful for improving the performance (Minka and Qi, 2004). Consider inference on the square grid as a problem where one can introduce tractable substructures without getting a very large increase in complexity. The spanning tree treats approximately half of the links exactly, whereas covering the grid with strips of width L would treat a fraction of $1 - 1/2L$ of the links exactly at a computational increase of a factor of 2^{L-1} compared to the spanning tree for the binary part, but keeping

Problem type			Method							
Graph	Coupling	d_{coup}	SP	LD	EC factorized			EC tree		
			Mean	Mean	Mean \pm std	Med	Max	Mean \pm std	Med	Max
Full	Repulsive	0.25	0.037	0.020	0.003 \pm 0.002	0.002	0.00	0.0017 \pm 0.0011	0.001	0.01
	Repulsive	0.50	0.071	0.018	0.031 \pm 0.045	0.016	0.20	0.0143 \pm 0.0141	0.010	0.10
	Mixed	0.25	0.004	0.020	0.002 \pm 0.002	0.002	0.00	0.0013 \pm 0.0008	0.001	0.00
	Mixed	0.50	0.055	0.021	0.022 \pm 0.030	0.013	0.17	0.0151 \pm 0.0204	0.010	0.16
	Attractive	0.06	0.024	0.027	0.004 \pm 0.002	0.004	0.01	0.0025 \pm 0.0014	0.002	0.01
	Attractive	0.12	0.435	0.033	0.117 \pm 0.090	0.112	0.30	0.0211 \pm 0.0307	0.012	0.16
Grid	Repulsive	1.0	0.294	0.047	0.153 \pm 0.123	0.124	0.58	0.0031 \pm 0.0021	0.003	0.01
	Repulsive	2.0	0.342	0.041	0.198 \pm 0.135	0.214	0.49	0.0021 \pm 0.0010	0.002	0.01
	Mixed	1.0	0.014	0.016	0.011 \pm 0.010	0.009	0.08	0.0018 \pm 0.0011	0.002	0.01
	Mixed	2.0	0.095	0.038	0.082 \pm 0.081	0.034	0.32	0.0068 \pm 0.0053	0.005	0.03
	Attractive	1.0	0.440	0.047	0.125 \pm 0.104	0.068	0.36	0.0028 \pm 0.0018	0.002	0.01
	Attractive	2.0	0.520	0.042	0.177 \pm 0.125	0.198	0.41	0.0002 \pm 0.0004	0.000	0.00

Table 1: The average one-norm error on marginals for the Wainwright-Jordan set-up.

the complexity of the most computationally expensive part of the inference—calculating the moments of the Gaussian part—unchanged.

A generalization of our method to treat graphical models beyond pair-wise interaction may be obtained by iterating the approximation. This is useful in cases, where an initial three term approximation $-\ln Z^{EC} = -\ln Z_q - \ln Z_r + \ln Z_s$ still contains non-tractable component free energies. These individual terms can be further approximated using the EC approach. We can show that in such a way a variety of other relevant types of graphical models beyond the pair-wise interaction case (on certain directed graphs and mixture models) become tractable with our method.

For practical applicability of approximate inference techniques improvements in the numerical implementation of the free energy minimization are crucial. In the simulations in this paper we used both single and double loop algorithms. The single loop algorithms often converged very fast, i.e. in $\mathcal{O}(10)$ iterations to achieve a solution close to the machine precision. However, whether convergence could be achieved was instance dependent and depended upon set-up details like parallel/sequential update and damping factor. It seems that there is a lot of room for improvement here and theoretical analysis of convergence properties of algorithms will be important in this respect (Heskes and Zoeter, 2002). In the guaranteed convergent double loop approaches the free energy minimization is formulated in terms of a sequence of convex optimization problems. This allows for the application of theoretically well-founded and powerful techniques of convex optimization (Boyd and Vandenberghe, 2004). Unfortunately, for the problems considered here, convergence is typically quite slow because we have to solve large number of the convex problems. This again underlines the need for further algorithmic development.

There are a couple of ways to improve on the EC approximation itself. One may calculate corrections to the EC free energy and marginals by a perturbative analysis using cumulant expansions of the approximating distributions. This should also enable a kind of sanity check of the theory, i.e. when the corrections are predicted to be comparable to original prediction,

it is a signal that the approximation is breaking down. Another possible improvement could come from physics of disordered system where methods have be devised to analyze non-ergodic free energy landscapes (Mézard et al., 1987). This will allow to make improved estimates of the free energy and marginals for example binary variables with large coupling strengths.

Acknowledgments

Discussions with and suggestions by Kees Albers, Bert Kappen, Tom Minka, Wim Wiegnerinck, Onno Zoeter and anonymous referees are greatly appreciated. Special thanks to Wim for his contributions to clarifying the single loop algorithm concepts.

Appendix A. Applications

In this appendix we give list of of previous applications of the ADATAP method which is a special case of the EC approach to models with the factorization eqs. (6) and (7).

Application	meaning of x_i	type of x_i	Refs.
Channel Division Multiple Access (CDMA)	source symbol	Ising	a
Gaussian Processes (GP) classification	latent variable	continuous	b
GP for wind retrieval	wind vector	continuous	c
Bootstrap estimates	latent variable	continuous	d
Independent component analysis (ICA)	source variable	arbitrary	e
Sparse kernel method	latent variable	continuous	f

Table 2: Examples of applications of simplest version of EC, ADATAP. The references are a: Fabricius and Winther (2004), b: Oppper and Winther (1999, 2000); Minka (2001a,b), c: Cornford et al. (2004), d: Malzahn and Oppper (2003, 2004), e: Hojen-Sorensen et al. (2002) and f: Quiñonero-Candela and Winther (2003).

Appendix B. Dual Formulation

In this appendix we present an alternative route to EC free energy approximation using a two stage variational formulation. The result is the so-called Gibbs free energy which is the Lagragian dual of the Helmholtz free energy eq. (19).

B.1 Gibbs Free Energies and Two Stage Inference

In this framework, one starts with the well known fact that the true, intractable distribution $p(\mathbf{x}) = \frac{f(\mathbf{x})}{Z}$ is implicitly characterized as the solution of an optimization problem defined through the relative entropy or KL divergence

$$KL(q, p) = \int d\mathbf{x} q(\mathbf{x}) \ln \frac{q(\mathbf{x})}{p(\mathbf{x})} \tag{38}$$

between p and other trial or approximate distributions q . We introduce the Gibbs free energy (GFE) approach, (see, e.g. Roepstorff, 1994; Csató et al., 2002; Wainwright and Jordan, 2003, 2005) which splits this optimization into a two stage process. One first constrains the trial distributions q by fixing the values of the generalized moments $\langle \mathbf{g}(\mathbf{x}) \rangle_q$. We define the Gibbs free energy $G(\boldsymbol{\mu})$ as

$$G(\boldsymbol{\mu}) = \min_q \{KL(q, p) \mid \langle \mathbf{g}(\mathbf{x}) \rangle_q = \boldsymbol{\mu}\} - \ln Z . \quad (39)$$

The term $\ln Z$ has been subtracted to make the resulting expression independent of the intractable partition function Z .

In a second step, the moments of the distribution and also the partition function Z are found within the same approach by relaxing the constraints and further minimizing $G(\boldsymbol{\mu})$ with respect to the $\boldsymbol{\mu}$.

$$\min_{\boldsymbol{\mu}} G(\boldsymbol{\mu}) = -\ln Z \quad (40)$$

$$\langle \mathbf{g}(\mathbf{x}) \rangle = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} G(\boldsymbol{\mu}) . \quad (41)$$

A variational bound approximation is recovered by restricting the minimization in eq. (39) to a tractable family of densities q . Note that the values for $\boldsymbol{\mu}$ in the definition of $G(\boldsymbol{\mu})$ cannot be chosen arbitrarily. For a detailed discussion of this problem, see Wainwright and Jordan (2003, 2005). We will not discuss these constraints here, but leave this, when necessary, to the discussion of concrete models.

Gibbs free energy and duality. The optimization problem eq. (39) is solved by the density given by

$$q(\mathbf{x}) = \frac{f(\mathbf{x})}{Z(\boldsymbol{\lambda})} \exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})) . \quad (42)$$

$\boldsymbol{\lambda} = \boldsymbol{\lambda}(\boldsymbol{\mu})$ is the vector of *Lagrange parameters* chosen such that the moment conditions $\langle \mathbf{g}(\mathbf{x}) \rangle_q = \boldsymbol{\mu}$ are fulfilled, i.e. $\boldsymbol{\lambda}$ satisfies

$$\frac{\partial \ln Z(\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = \boldsymbol{\mu} . \quad (43)$$

In the following, it should be clear from the context when $\boldsymbol{\lambda}$ is a free variable or is to be determined from eq. (43). Inserting the optimizing distribution eq. (42) into the definition of the Gibbs free energy eq. (39), we get the simpler expression:

$$G(\boldsymbol{\mu}) = -\ln Z(\boldsymbol{\lambda}(\boldsymbol{\mu})) + \boldsymbol{\lambda}^T(\boldsymbol{\mu})\boldsymbol{\mu} = \max_{\boldsymbol{\lambda}} \{-\ln Z(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T \boldsymbol{\mu}\} . \quad (44)$$

showing that $G(\boldsymbol{\mu})$ is the Lagrangian dual of $\ln Z(\boldsymbol{\lambda})$.

Derivatives with respect to parameters. We will use the following result about the derivative of G with respect to a parameter t in the density. Using the notation $p(\mathbf{x}|t) = \frac{f(\mathbf{x},t)}{Z_t}$ (which should not be confused with a conditional probability), we calculate the derivative of $G(\boldsymbol{\mu}, t)$ using (43) and (44) as for fixed $\boldsymbol{\mu}$:

$$\frac{dG(\boldsymbol{\mu}, t)}{dt} = -\frac{\partial \ln Z(\boldsymbol{\lambda}, t)}{\partial t} + \left(\boldsymbol{\mu} - \frac{\partial \ln Z(\boldsymbol{\lambda}, t)}{\partial \boldsymbol{\lambda}} \right) \frac{d\boldsymbol{\lambda}^T}{dt} = -\frac{\partial \ln Z(\boldsymbol{\lambda}, t)}{\partial t} , \quad (45)$$

where $Z(\boldsymbol{\lambda}, t) = \int d\mathbf{x} f(\mathbf{x}, t) \exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}))$.

B.2 An Interpolation Representation of Free Energies

If the density p factors into a tractable f_q and an intractable part f_r , according to eq. (2), we can construct a representation of the Gibbs free energy which also separates into two corresponding parts. This is done by treating $f_r(\mathbf{x})$ as a *perturbation* which is smoothly turned on using a parameter $0 \leq t \leq 1$. We define $f_r(\mathbf{x}, t)$ to be a smooth interpolation between the trivial $f_r(\mathbf{x}, t = 0) = 1$ and the “full” intractable $f_r(\mathbf{x}, t = 1) = f_r(\mathbf{x})$. The most common choice is to set $f_r(\mathbf{x}, t) = [f_r(\mathbf{x})]^t$, but a more complicated construction can be necessary, when f_r contains δ -distributions, see appendix E. However, we will see later, that an explicit construction of the interpolation will not be necessary for our approximation.

Next, we define the interpolating density and the associated optimizing distribution for the Gibbs free energy

$$p(\mathbf{x}|t) = \frac{1}{Z_t} f_q(\mathbf{x}) f_r(\mathbf{x}, t) \tag{46}$$

$$q(\mathbf{x}|t) = \frac{1}{Z_q(\boldsymbol{\lambda}, t)} f_q(\mathbf{x}) f_r(\mathbf{x}, t) \exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})) \ , \tag{47}$$

where

$$Z_q(\boldsymbol{\lambda}, t) = \int d\mathbf{x} f_q(\mathbf{x}) f_r(\mathbf{x}, t) \exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})) \tag{48}$$

and the corresponding free energy $G_q(\boldsymbol{\mu}, t) = \max_{\boldsymbol{\lambda}} \{-\ln Z_q(\boldsymbol{\lambda}, t) + \boldsymbol{\lambda}^T \boldsymbol{\mu}\}$. For later convenience, we have given a subscript to G and $\ln Z$ to indicate which approximating distribution is being used. We can now use the following simple identity for the free energy $G(\boldsymbol{\mu}, t)$

$$G(\boldsymbol{\mu}, 1) - G(\boldsymbol{\mu}, 0) = \int_0^1 dt \frac{dG(\boldsymbol{\mu}, t)}{dt} \tag{49}$$

to relate the Gibbs free energy of the intractable model $G(\boldsymbol{\mu}) = G(\boldsymbol{\mu}, t = 1)$ and tractable model $G(\boldsymbol{\mu}, t = 0)$. Using eq. (20), we get

$$\frac{dG(\boldsymbol{\mu}, t)}{dt} = -\frac{\partial \ln Z(\boldsymbol{\lambda}, t)}{\partial t} = -\left\langle \frac{d \ln f_r(\mathbf{x}, t)}{dt} \right\rangle_{q(\mathbf{x}|t)} \ . \tag{50}$$

While this representation can be used to re-derive a variational bound approximation (see Appendix F), we will next re-derive a dual representation of the EC free energy by making an approximation similar in spirit to the one used in Section 3. We again assume that besides the family of distributions eq. (4), there is a second family which can be used as an approximation to the distribution eq. (46). It is defined by

$$r(\mathbf{x}|t) = \frac{1}{Z_r(\boldsymbol{\lambda}, t)} f_r(\mathbf{x}, t) \exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})) \ , \tag{51}$$

where, as before the parameters $\boldsymbol{\lambda}$ are chosen in such a way as to guarantee *consistency for the expectations* of \mathbf{g} , i.e. $\langle \mathbf{g}(\mathbf{x}) \rangle_{r(\mathbf{x}|t)} = \boldsymbol{\mu}$ and

$$Z_r(\boldsymbol{\lambda}, t) = \int d\mathbf{x} f_r(\mathbf{x}, t) \exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})) \ . \tag{52}$$

Obviously, $r(\mathbf{x}|t)$ defines another Gibbs free energy which in its dual representation eq. (44) is given by

$$G_r(\boldsymbol{\mu}, t) = \max_{\boldsymbol{\lambda}} \left\{ -\ln Z_r(\boldsymbol{\lambda}, t) + \boldsymbol{\lambda}^T \boldsymbol{\mu} \right\} . \quad (53)$$

Using the density $r(\mathbf{x}|t)$ to treat the integral in eq. (49), we make the approximation

$$\int_0^1 dt \left\langle \frac{d \ln f_r(\mathbf{x}, t)}{dt} \right\rangle_{q(\mathbf{x}|t)} \approx \int_0^1 dt \left\langle \frac{d \ln f_r(\mathbf{x}, t)}{dt} \right\rangle_{r(\mathbf{x}|t)} . \quad (54)$$

The fact that both types of densities eqs. (47) and (51) contain the same exponential factor $f_r(\mathbf{x}, t) \exp(\boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}))$ allows us to carry out the integral over the interaction strength t on the right hand side of eq. (54) in closed form without specifying the interpolating term $f_r(\mathbf{x}, t)$ explicitly. We simply use the relations eqs. (49) and (50) again, but this time for the free energy eq. (53) to get

$$\int_0^1 dt \left\langle \frac{d \ln f_r(\mathbf{x}, t)}{dt} \right\rangle_{r(\mathbf{x}|t)} = G_r(\boldsymbol{\mu}, 1) - G_r(\boldsymbol{\mu}, 0) . \quad (55)$$

Using the approximation eq. (54) and the two exact relation eqs. (49) for q and r we arrive at the *expectation consistent (EC)* approximation:

$$G_q(\boldsymbol{\mu}, 1) \approx G_q(\boldsymbol{\mu}, 0) + G_r(\boldsymbol{\mu}, 1) - G_r(\boldsymbol{\mu}, 0) \equiv G^{\text{EC}}(\boldsymbol{\mu}) . \quad (56)$$

Recovering the EC free energy eq. (19) Using the duality expression for the free energies eq. (44), the free energy approximation can be written as

$$\begin{aligned} G^{\text{EC}}(\boldsymbol{\mu}) &= G_q(\boldsymbol{\mu}) + G_r(\boldsymbol{\mu}) - G_s(\boldsymbol{\mu}) \\ &= \max_{\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_r} \min_{\boldsymbol{\lambda}_s} \left\{ -\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_r) + \ln Z_s(\boldsymbol{\lambda}_s) + \boldsymbol{\mu}^T (\boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r - \boldsymbol{\lambda}_s) \right\} , \end{aligned} \quad (57)$$

where we have defined $G_q(\boldsymbol{\mu}) = G_q(\boldsymbol{\mu}, 0)$, $G_r(\boldsymbol{\mu}) = G_r(\boldsymbol{\mu}, 1)$ and $G_s(\boldsymbol{\mu}) = G_r(\boldsymbol{\mu}, 0)$. To obtain the corresponding approximation for the Helmholtz free energy $-\ln Z$, we should minimize this expression with respect to $\boldsymbol{\mu}$. Any local minimum will be characterized by the vanishing of the partial derivative with respect to $\boldsymbol{\mu}$. This yields the following constraint on the Lagrange parameters

$$\boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r - \boldsymbol{\lambda}_s = 0 , \quad (58)$$

which can be used to eliminate, say $\boldsymbol{\lambda}_r$ and we recover eq. (19).

Recovering the double loop algorithm. Since the free energy given by eq. (44) is a convex function of $\boldsymbol{\mu}$, we can see that the EC approximation eq. (56) appears directly as a sum of a convex (the first two terms) and a concave function of $\boldsymbol{\mu}$. Hence, the approximation is not guaranteed to be convex, and multiple local minima and other stationary points may occur. However, this natural split allows us to develop a double loop algorithm similar to Yuille (2002); Heskes et al. (2003), which is guaranteed to converge to at least one of the stationary points, provided that the EC free energy is bounded from below. Assume that at iteration step t , the current approximation to the minimizer $\boldsymbol{\mu}(t)$,

such an algorithm first upper bounds the concave function $-G_s(\boldsymbol{\mu})$ by the linear function $-(\boldsymbol{\mu} - \boldsymbol{\mu}(t))^T \frac{\partial G_s(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} \Big|_{\boldsymbol{\mu}=\boldsymbol{\mu}(t)}$.

In terms of the corresponding Lagrange-parameter $\boldsymbol{\lambda}_s(t) = \frac{\partial G_s(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} \Big|_{\boldsymbol{\mu}=\boldsymbol{\mu}(t)}$, this yields

$$\begin{aligned} G^{\text{EC}}(\boldsymbol{\mu}) &\leq G_q(\boldsymbol{\mu}) + G_r(\boldsymbol{\mu}) - (\boldsymbol{\mu} - \boldsymbol{\mu}(t))^T \boldsymbol{\lambda}_s(t) \\ &= \max_{\boldsymbol{\lambda}_q, \boldsymbol{\lambda}_r} \{-\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_r) + \boldsymbol{\mu}^T(\boldsymbol{\lambda}_q + \boldsymbol{\lambda}_r) + \ln Z_s(\boldsymbol{\lambda}_s(t))\} \equiv G_t^{\text{EC}}(\boldsymbol{\mu}) \end{aligned}$$

Minimizing $G_t^{\text{EC}}(\boldsymbol{\mu})$ with respect to $\boldsymbol{\mu}$, we immediately get

$$\min_{\boldsymbol{\mu}} G_t^{\text{EC}}(\boldsymbol{\mu}) = \max_{\boldsymbol{\lambda}_q} \{-\ln Z_q(\boldsymbol{\lambda}_q) - \ln Z_r(\boldsymbol{\lambda}_s - \boldsymbol{\lambda}_q)\} + \ln Z_s(\boldsymbol{\lambda}_s(t)) = F(\boldsymbol{\lambda}_s(t)) , \quad (59)$$

where $F(\boldsymbol{\lambda}_s(t))$ was introduced in eq. (26). The new approximation is computed as

$$\boldsymbol{\mu}(t+1) = \langle \mathbf{g}(\mathbf{x}) \rangle_{q(t+1)} .$$

Hence, this double loop procedure is equivalent to the one defined in Section 5, demonstrating that the sequence $F(\boldsymbol{\lambda}_s(t))$ yields nondecreasing upper bounds to the minimal EC Gibbs free energy.

Appendix C. Tree-Connected Graphs

For the EC tree approximation we will need to make inference on tree-connected graphs. To handle a problem with binary variables both binary and Gaussian distributed variables on a tree will be needed. We will write the model as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_i \psi_i(\mathbf{x}_i) \exp \left(-\frac{1}{2} \mathbf{x}^T \boldsymbol{\Lambda} \mathbf{x} + \boldsymbol{\gamma}^T \mathbf{x} \right) ,$$

where $\psi_i(x_i) = \delta(x_i - 1) + \delta(x_i + 1)$ for binary and $\psi_i(x_i) = 1$ for Gaussian. Assuming that $\boldsymbol{\Lambda}$ defines a tree one can express the free energy in terms of single- and two-node free energies (Yedidia et al., 2001):

$$-\ln Z(\boldsymbol{\lambda}) = - \sum_{(ij) \in \mathcal{G}} \ln Z_{ij}(\boldsymbol{\lambda}^{(ij)}) - \sum_i (1 - n_i) \ln Z_i(\boldsymbol{\lambda}^{(i)}) , \quad (60)$$

where $\boldsymbol{\lambda}^{(ij)} = \left(\gamma_i^{(ij)}, \gamma_j^{(ij)}, \Lambda_{ii}^{(ij)}, \Lambda_{ij}^{(ij)}, \Lambda_{jj}^{(ij)} \right)$ are the parameters associated with the moments $\mathbf{g}^{(ij)} = \left(x_i, x_j, -\frac{x_i^2}{2}, -x_i x_j, -\frac{x_j^2}{2} \right)$ and n_i is the number of links to node i . The two-node partition function Z_{ij} is given by

$$Z_{ij}(\boldsymbol{\lambda}^{(ij)}) = \int dx_i dx_j \psi_i(x_i) \psi_j(x_j) e^{\gamma_i x_i + \gamma_j x_j - \Lambda_{ij} x_i x_j - \Lambda_{ii} x_i^2 / 2 - \Lambda_{jj} x_j^2 / 2} . \quad (61)$$

The one-node partition function is defined in a similar fashion.

The Gibbs free energy $G(\boldsymbol{\mu}) = \max_{\boldsymbol{\lambda}} \{-\ln Z(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T \boldsymbol{\mu}\}$ can be written in terms of one- and two-node Gibbs free energies:

$$\begin{aligned} G(\boldsymbol{\mu}) &= \sum_{(ij) \in \mathcal{G}} \ln G_{ij}(\boldsymbol{\mu}^{(ij)}) - \sum_i (1 - n_i) G_i(\boldsymbol{\mu}^{(i)}) \\ G_{ij}(\boldsymbol{\mu}^{(ij)}) &= \max_{\boldsymbol{\lambda}^{(ij)}} \{-\ln Z_{ij}(\boldsymbol{\lambda}^{(ij)}) + (\boldsymbol{\lambda}^{(ij)})^T \boldsymbol{\mu}^{(ij)}\}, \end{aligned} \quad (62)$$

where $\boldsymbol{\mu}^{(ij)} = \langle \mathbf{g}^{(ij)}(\mathbf{x}) \rangle$. We can write $\boldsymbol{\lambda} = \sum_{(ij) \in \mathcal{G}} \boldsymbol{\lambda}^{(ij)} - \sum_i (1 - n_i) \boldsymbol{\lambda}^{(i)}$, where $\boldsymbol{\lambda}^{(ij)}$ here should be understood as a vector of the same length as \mathbf{g} having non-zero elements for moments defined for the pair (ij) . By solving the max condition we can write the Lagrange parameters in terms of the mean values $m_i = \langle x_i \rangle$ and covariances $\chi_{ij} = \langle x_i x_j \rangle - m_i m_j$. This will be useful when we derive algorithms for optimizing the free energy in section 5 where we need to solve for $\boldsymbol{\lambda}$ in terms of $\boldsymbol{\mu}$. For binary variables we get:

$$\begin{aligned} \gamma_i^{(i)} &= \tanh^{-1}(m_i) \\ \gamma_i^{(ij)} &= \frac{1}{2} \tanh^{-1} \left(\frac{m_i + m_j}{1 + \langle x_i x_j \rangle} \right) + \frac{1}{2} \tanh^{-1} \left(\frac{m_i - m_j}{1 - \langle x_i x_j \rangle} \right) \\ \gamma_j^{(ij)} &= \frac{1}{2} \tanh^{-1} \left(\frac{m_i + m_j}{1 + \langle x_i x_j \rangle} \right) + \frac{1}{2} \tanh^{-1} \left(\frac{m_j - m_i}{1 - \langle x_i x_j \rangle} \right) \\ \Lambda_{ij}^{(ij)} &= -\frac{1}{2} \tanh^{-1} \left(\frac{\langle x_i x_j \rangle + m_i}{1 + m_j} \right) - \frac{1}{2} \tanh^{-1} \left(\frac{\langle x_i x_j \rangle - m_i}{1 - m_j} \right) \end{aligned}$$

and for Gaussian defining $\mathbf{m}^{(ij)} = \begin{pmatrix} m_i \\ m_j \end{pmatrix}$ and $\boldsymbol{\chi}^{(ij)} \equiv \begin{pmatrix} \chi_{ii} & \chi_{ij} \\ \chi_{ji} & \chi_{jj} \end{pmatrix}$:

$$\begin{aligned} \gamma_i^{(i)} &= m_i / \chi_{ii} & \text{and} & & \Lambda_i^{(i)} &= 1 / \chi_{ii} \\ \gamma^{(ij)} &= (\boldsymbol{\chi}^{(ij)})^{-1} \mathbf{m}^{(ij)} & \text{and} & & \boldsymbol{\Lambda}^{(ij)} &= (\boldsymbol{\chi}^{(ij)})^{-1}. \end{aligned}$$

Finally, we will also need to make inference about the mean values and covariances on the tree for the binary variables. This can be done effectively by message passing on the tree. The message from link (ij) to node i denoted by $r_{(ij) \rightarrow i}$ can be obtained by the following recursion (MacKay, 2003)

$$\begin{aligned} r_{(ij) \rightarrow i} &= \tanh(-\Lambda_{ij}) \tanh(\theta_{j \setminus i}) \\ \theta_{j \setminus i} &= \theta_j + \sum_{k, (jk) \in \mathcal{G}, (jk) \neq (ij)} r_{(jk) \rightarrow j}. \end{aligned}$$

The recursion converges in one collect and one distribute messages sweep (to/from an arbitrarily chosen root node). Inference is linear because the tree contains $N - 1$ links. The mean values and correlations are given by

$$\begin{aligned} m_i &= \tanh \left(\theta_i + \sum_{j, (ij) \in \mathcal{G}} r_{(ij) \rightarrow i} \right) \\ \langle x_i x_j \rangle &= \frac{e^{-\Lambda_{ij}} \cosh(\theta_{i \setminus j} + \theta_{j \setminus i}) - e^{\Lambda_{ij}} \cosh(\theta_{i \setminus j} - \theta_{j \setminus i})}{e^{-\Lambda_{ij}} \cosh(\theta_{i \setminus j} + \theta_{j \setminus i}) + e^{\Lambda_{ij}} \cosh(\theta_{i \setminus j} - \theta_{j \setminus i})}. \end{aligned}$$

Appendix D. Single Loop Algorithmic Recipes

In this appendix we give the algorithmic recipes for one sequential algorithm for the factorized EC and a parallel algorithm for tree EC. The sequential algorithm is close in spirit to Expectation Propagation with $\psi_i(x_i)$ and $\exp(\gamma_{r,i}x_i - \frac{1}{2}\Lambda_{r,i}x_i^2)$ being what is called exact and approximate factors, respectively (Minka, 2001b):

- Initialize mean and covariance of r -distribution:

$$\begin{aligned}\mathbf{m}_r &:= (\mathbf{\Lambda}_r - \mathbf{J})^{-1}(\boldsymbol{\gamma}_r + \boldsymbol{\theta}) \\ \boldsymbol{\chi}_r &:= (\mathbf{\Lambda}_r - \mathbf{J})^{-1}\end{aligned}$$

with $\boldsymbol{\gamma}_r = \mathbf{0}$ and $\mathbf{\Lambda}_r$ set such that the covariance is positive definite.

Run sequentially over the nodes:

1. Send message from r to q_i
 - Calculate separator s_i : $\gamma_{s,i} := m_{r,i}/\chi_{r,ii}$ and $\Lambda_{s,i} := 1/\chi_{r,ii}$.
 - Update q_i : $\gamma_{q,i} := \gamma_{s,i} - \gamma_{r,i}$ and $\Lambda_{q,i} := \Lambda_{s,i} - \Lambda_{r,i}$.
 - Update moments of q_i : $m_{q,i} := \tanh(\gamma_{q,i})$ and $\chi_{q,ii} = 1 - m_{q,i}^2$.
2. Send message from q_i to r
 - Calculate separator s_i : $\gamma_{s,i} := m_{q,i}/\chi_{q,ii}$ and $\Lambda_{s,i} := 1/\chi_{q,ii}$.
 - Update r : $\gamma_{r,i} := \gamma_{s,i} - \gamma_{q,i}$, $\Delta\Lambda_{r,i} := \Lambda_{s,i} - \Lambda_{q,i} - \Lambda_{r,i}$ and $\Lambda_{r,i} := \Lambda_{s,i} - \Lambda_{q,i}$.
 - Update moments of r (see eq. 37):

$$\begin{aligned}\boldsymbol{\chi}_r &:= \boldsymbol{\chi}_r - \frac{\Delta\Lambda_{r,i}}{1 + \Delta\Lambda_{r,i}[\boldsymbol{\chi}_r]_{ii}}[\boldsymbol{\chi}_r]_i[\boldsymbol{\chi}_r]_i^T \\ \mathbf{m}_r &:= \boldsymbol{\chi}_r(\boldsymbol{\gamma}_r + \boldsymbol{\theta}).\end{aligned}$$

Convergence is reached when and if $\mathbf{m}_r = \mathbf{m}_q$ and $\chi_{r,ii} = \chi_{q,ii}$, $i = 1, \dots, N$. The computational complexity of the algorithm is $\mathcal{O}(N^3 N_{\text{ite}})$ because each Sherman-Morrison update is $\mathcal{O}(N^2)$ and we make N of those in each sweep over the nodes.

The tree EC algorithm is very similar. The only difference is that it is parallel and uses inference on a tree graph, see appendix C for details on the tree inference:

- Initialize as above.

Update:

1. Send message from r to q
 - Calculate separator s : $[\boldsymbol{\gamma}_s, \mathbf{\Lambda}_s] := \text{Lagrange_Gauss_tree}(\mathbf{m}_r, \text{tree}(\boldsymbol{\chi}_r))$, where $\text{tree}()$ sets all non-tree elements to zero.
 - Update q : $\boldsymbol{\gamma}_q := \boldsymbol{\gamma}_s - \boldsymbol{\gamma}_r$ and $\mathbf{\Lambda}_q := \mathbf{\Lambda}_s - \mathbf{\Lambda}_r$.
 - Update moments of q : $[m_q, \boldsymbol{\chi}_q] := \text{inference_binary_tree}(\boldsymbol{\gamma}_q, \mathbf{\Lambda}_q)$ will only return non-zero elements of the covariance on the tree.

2. Send message from q to r

- Calculate separator s : $[\gamma_s, \Lambda_s] := \text{Lagrange_Gauss_tree}(\mathbf{m}_q, \chi_q)$.
- Update r : $\gamma_r := \gamma_s - \gamma_q$ and $\Lambda_r := \Lambda_s - \Lambda_q$.
- Update moments of r : $\chi_r := (\Lambda_r - \mathbf{J})^{-1}$ and $\mathbf{m}_r := \chi_r(\gamma_r + \theta)$.

Convergence is reached when $\mathbf{m}_q = \mathbf{m}_r$ and $\chi_q = \text{tree}(\chi_r)$. This algorithm is also $\mathcal{O}(N^3 N_{\text{ite}})$ because of the matrix inverse. All other operations are $\mathcal{O}(N)$ even though these will dominate for small N . Typically when convergent both algorithms converge in $N_{\text{ite}} = \mathcal{O}(10)$ steps.

Appendix E. Interpolation Scheme for Discrete Variables

The Ising case eq. (9) can be treated by defining the bimodal density

$$f_r(\mathbf{x}, t) = \prod_{i=1}^N \left(\frac{\exp \left[-\frac{t}{1-t} (x_i^4 - 2x_i^2) \right]}{\sqrt{1-t}} \right)$$

which interpolates between a constant function for $t = 0$ and becomes proportional to the Dirac measures eq. (9) in the limit $t \rightarrow 1$. Other discrete variables can be treated in a similar fashion.

Appendix F. Re-deriving the Variational Bound Approximation

The choice $f_r(\mathbf{x}, t) = t \ln f_r(\mathbf{x})$ for the interpolation can be used for a perturbation expansion of the free energy $G(\boldsymbol{\mu}, t)$ in powers of t , where at the end one sets $t = 1$. The lowest nontrivial (first) order term is obtained by replacing $q(\mathbf{x}|t)$ by $q(\mathbf{x}|0)$ in eq. (50). In this case, one obtains an approximation to the Gibbs free energy given by

$$G(\boldsymbol{\mu}) \approx G(\boldsymbol{\mu}, 0) - \int_0^1 dt \left\langle \frac{d \ln f_r(\mathbf{x}, t)}{dt} \right\rangle_{q(\mathbf{x}|0)} = G(\boldsymbol{\mu}, 0) - \langle \ln f_r(\mathbf{x}) \rangle_{q(\mathbf{x}|0)}. \quad (63)$$

For the second order term of this so-called Plefka expansion see, e.g. Plefka (1982) and several contributions in Opper and Saad (2001).

For comparison, we define a variational bound approximation, where the minimization in eq. (39) is restricted to the family \mathcal{F} of densities of the form eq. (4), i.e.

$$G^{\text{var}}(\boldsymbol{\mu}) = \min_{q \in \mathcal{F}} \{ KL(q, p) \mid \langle \mathbf{g}(\mathbf{x}) \rangle_q = \boldsymbol{\mu} \} - \ln Z. \quad (64)$$

Since we are minimizing in a restricted class of distributions, we obtain the upper bound $G(\boldsymbol{\mu}) \leq G^{\text{var}}(\boldsymbol{\mu})$ on the Gibbs free energy. Using the fact that the density eq. (4) is exactly of the form of $q(\mathbf{x}|0)$, we can show that $G^{\text{var}}(\boldsymbol{\mu})$ coincides exactly with eq. (63).

References

- H. Attias. A variational Bayesian framework for graphical models. In T. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press, 2000.
- C. M. Bishop, D. Spiegelhalter, and J. Winn. Vibes: A variational inference engine for Bayesian networks. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 777–784. MIT Press, 2003.
- S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- D. Cornford, L. Csató, D. J. Evans, and M. Opper. Bayesian analysis of the scatterometer wind retrieval inverse problem: Some new approaches. *Journal Royal Statistical Society B*, 66:1–17, 2004.
- L. Csató, M. Opper, and O. Winther. TAP Gibbs free energy, belief propagation and sparsity. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 657–663, Cambridge, MA, 2002. MIT Press.
- T. Fabricius and O. Winther. Correcting the bias of subtractive interference cancellation in cdma: Advanced mean field theory. *Submitted to IEEE trans. Inf. Theory*, 2004.
- T. Heskes, K. Albers, and H. Kappen. Approximate inference and constrained optimization. In *Proceedings UAI-2003*, pages 313–320. Morgan Kaufmann, 2003.
- T. Heskes and O. Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In A. Darwiche and N. Friedman, editors, *Proceedings UAI-2002*, pages 216–233, 2002.
- P. A.d.F.R. Hojen-Sorensen, O. Winther, and L. K. Hansen. Mean field approaches to independent component analysis. *Neural Computation*, 14:889–918, 2002.
- Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37:183–233, 1999.
- D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- D. Malzahn and M. Opper. An approximate analytical approach to resampling averages. *Journal of Machine Learning Research*, pages 1151–1173, 2003.
- D. Malzahn and M. Opper. Approximate analytical bootstrap averages for support vector classifiers. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.
- M. Mézard, G. Parisi, and M. A. Virasoro. *Spin Glass Theory and Beyond*, volume 9 of *Lecture Notes in Physics*. World Scientific, 1987.

- T. P. Minka. Expectation propagation for approximate Bayesian inference. In J. S. Breese and D. Koller, editors, *Proceedings UAI-2001*, pages 362–369. Morgan Kaufmann, 2001a.
- T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT Media Lab, 2001b.
- T. P. Minka and Y. Qi. Tree-structured approximations by expectation propagation. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.
- M. Opper and D. Saad, editors. *Advanced Mean Field Methods: Theory and Practice*. MIT Press, 2001.
- M. Opper and O. Winther. Mean field methods for classification with gaussian processes. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 309–315. MIT Press, 1999.
- M. Opper and O. Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12:2655–2684, 2000.
- M. Opper and O. Winther. Adaptive and self-averaging Thouless-Anderson-Palmer mean field theory for probabilistic modeling. *Phys. Rev. E*, 64:056131, 2001a.
- M. Opper and O. Winther. Tractable approximations for probabilistic models: The adaptive Thouless-Anderson-Palmer mean field approach. *Phys. Rev. Lett.*, 86:3695, 2001b.
- M. Opper and O. Winther. Variational linear response. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
- T. Plefka. Convergence condition of the TAP equation for the infinite-range Ising spin glass. *J. Phys. A*, 15:1971, 1982.
- J. Quiñero-Candela and O. Winther. Incremental gaussian processes. In S. Thrun, S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1001–1008. MIT Press, 2003.
- G. Roepstorff. *Path Integral Approach to Quantum Physics, An Introduction*. Springer - Verlag Berlin Heidelberg, New York, 1994.
- M. Suzuki, editor. *Coherent Anomaly Method, Mean Field, Fluctuations and Symmetries*. World Scientific, 1995.
- L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19:499–533, 1998.
- M. J. Wainwright and M. I. Jordan. Semidefinite methods for approximate inference on graphs with cycles. Technical Report UCB/CSD-03-1226, UC Berkeley CS Division, 2003.

- M. J. Wainwright and M. I. Jordan. A variational principle for graphical models. In S. Haykin, J. Principe, S. Sejnowski, and J McWhirter, editors, *New Directions in Statistical Signal Processing: From Systems to Brain*. MIT Press, 2005.
- M. Welling and Y.W. Teh. Approximate inference in Boltzmann machines. *Artificial Intelligence*, 143:19–50, 2003.
- J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 689–695, 2001.
- A. L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation. *Neural Computation*, 14:1691–1722, 2002.
- A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15: 915–936, 2003.