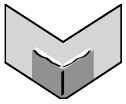
The Journal of Machine Learning Research
Volume 5 (2005)
Print-Archive Edition

Pages 801–1595



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 5 (2005)
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 late 2003 and 2004.

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/.

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

JMLR
Journal of Machine Learning Research Vol. 5 March 2005

1 **Learning Rates for Q-learning**
Eyal Even Dar, Yishay Mansour

27 **Learning the Kernel Matrix with Semidefinite Programming**
Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, Michael I. Jordan

73 **Dimensionality Reduction for Supervised Learning with Reproducing Kernel Hilbert Spaces**
Kenji Fukumizu, Francis R. Bach, Michael I. Jordan

101 **In Defense of One-Vs-All Classification**
Ryan Rifkin, Aldebaro Klautau

143 **Lossless Online Bayesian Bagging**
Herbert K. H. Lee, Merlise A. Clyde

153 **Subgroup Discovery with CN2-SD**
Nada Lavrač, Branko Kavšek, Peter Flach, Ljupčo Todorovski

189 **Generalization Error Bounds for Threshold Decision Lists**
Martin Anthony

219 **On the Importance of Small Coordinate Projections**
Shahar Mendelson, Petra Philips

239 **Weather Data Mining Using Independent Component Analysis**
Jayanta Basak, Anant Sudarshan, Deepak Trivedi, M. S. Santhanam

255 **Online Choice of Active Learning Algorithms**
Yoram Baram, Ran El Yaniv, Kobi Luz

293 **A Compression Approach to Support Vector Model Selection**
Ulrike von Luxburg, Olivier Bousquet, Bernhard Schölkopf

325 **A Geometric Approach to Multi-Criterion Reinforcement Learning**
Shie Mannor, Nahum Shimkin

361 **RCV1: A New Benchmark Collection for Text Categorization Research**
David D. Lewis, Yiming Yang, Tony G. Rose, Fan Li

399 **Distributional Scaling: An Algorithm for Structure-Preserving Embedding of Metric and Nonmetric Spaces**
Michael Quist, Golan Yona

421 **Learning Ensembles from Bites: A Scalable and Accurate Approach**
Nitesh V. Chawla, Lawrence O. Hall, Kevin W. Bowyer,W. Philip Kegelmeyer

453 **Robust Principal Component Analysis with Adaptive Selection for Tuning Parameters**
Isao Higuchi, Shinto Eguchi

473 **PAC-learnability of Probabilistic Deterministic Finite State Automata**
Alexander Clark, Franck Thollard

499 **Sources of Success for Boosted Wrapper Induction**
David Kauchak, Joseph Smarr, Charles Elkan

529 **Computable Shell Decomposition Bounds**
John Langford, David McAllester

549 **Exact Bayesian Structure Discovery in Bayesian Networks**
Mikko Koivisto, Kismat Sood

575 **A Universal Well-Calibrated Algorithm for On-line Classification (Special Topic on Learning Theory)**
Vladimir Vovk

605 **New Techniques for Disambiguation in Natural Language and Their Application to Biological Text**
Filip Ginter, Jorma Boberg, Jouni Järvinen, Tapio Salakoski

623 **The Sample Complexity of Exploration in the Multi-Armed Bandit Proble (Special Topic on Learning Theory)**
Shie Mannor, John N. Tsitsiklis

649 **Preference Elicitation and Query Learning (Special Topic on Learning Theory)**
Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, Martin Zinkevich

669 **Distance-Based Classification with Lipschitz Functions (Special Topic on Learning Theory)**
Ulrike von Luxburg, Olivier Bousquet

697 **Hierarchical Latent Class Models for Cluster Analysis**
Nevin L. Zhang

725 **Bias-Variance Analysis of Support Vector Machines for the Development of SVM-Based Ensemble Methods**
Giorgio Valentini, Thomas G. Dietterich

777 **A Fast Algorithm for Joint Diagonalization with Non-orthogonal Transformations and its Application to Blind Source Separation**
Andreas Ziehe, Pavel Laskov, Guido Nolte, Klaus-Robert Müller

801 **Feature Discovery in Non-Metric Pairwise Data**
Julian Laub, Klaus-Robert Müller

819 **Probability Product Kernels (Special Topic on Learning Theory)**
Tony Jebara, Risi Kondor, Andrew Howard

845 **Feature Selection for Unsupervised Learning**
Jennifer G. Dy, Carla E. Brodley

891 **Some Dichotomy Theorems for Neural Learning Problems**
Michael Schmitt

913 **Image Categorization by Learning and Reasoning with Regions**
Yixin Chen, James Z. Wang

941 **Boosting as a Regularized Path to a Maximum Margin Classifier**
Saharon Rosset, Ji Zhu, Trevor Hastie

975 **Probability Estimates for Multi-class Classification by Pairwise Coupling**
Ting-Fan Wu, Chih-Jen Lin, Ruby C. Weng

1007 **On Robustness Properties of Convex Risk Minimization Methods for Pattern Recognition**
Andreas Christmann, Ingo Steinwart

1035 **Rational Kernels: Theory and Algorithms (Special Topic on Learning Theory)**
Corinna Cortes, Patrick Haffner, Mehryar Mohri

1063 **Reinforcement Learning with Factored States and Actions**
Brian Sallans, Geoffrey E. Hinton

1089 **No Unbiased Estimator of the Variance of K-Fold Cross-Validation**
Yoshua Bengio, Yves Grandvalet

1107 **Selective Rademacher Penalization and Reduced Error Pruning of Decision Trees**
Matti Kääriäinen, Tuomo Malinen, Tapio Elomaa

1127 **Knowledge-Based Kernel Approximation**
Olvi L. Mangasarian, Jude W. Shavlik, Edward W. Wild

1143 **Support Vector Machine Soft Margin Classifiers: Error Analysis**
Di-Rong Chen, Qiang Wu, Yiming Ying, Ding-Xuan Zhou

1177 **Model Averaging for Prediction with Discrete Bayesian Networks**
Denver Dash, Gregory F. Cooper

1205 **Efficient Feature Selection via Analysis of Relevance and Redundancy**
Lei Yu, Huan Liu

1225 **Statistical Analysis of Some Multi-Category Large Margin Classification Methods**
Tong Zhang

1253 **The Minimum Error Minimax Probability Machine**
Kaizhu Huang, Haiqin Yang, Irwin King, Michael R. Lyu, Laiwan Chan

1287 Large-Sample Learning of Bayesian Networks is NP-Hard
David Maxwell Chickering, David Heckerman, Christopher Meek

1331 Randomized Variable Elimination
David J. Stracuzzi, Paul E. Utgoff

1363 Some Properties of Regularized Kernel Methods
Ernesto De Vito, Lorenzo Rosasco, Andrea Caponnetto, Michele Piana, Alessandro Verri

1391 The Entire Regularization Path for the Support Vector Machine
Trevor Hastie, Saharon Rosset, Robert Tibshirani, Ji Zhu

1417 Second Order Cone Programming Formulations for Feature Selection
Chiranjib Bhattacharyya

1435 Fast String Kernels using Inexact Matching for Protein Sequences
Christina Leslie, Rui Kuang

1457 Non-negative Matrix Factorization with Sparseness Constraints
Patrik O. Hoyer

1471 Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning
Evan Greensmith, Peter L. Bartlett, Jonathan Baxter

1531 Fast Binary Feature Selection with Conditional Mutual Information
François Fleuret

1557 The Dynamics of AdaBoost: Cyclic Behavior and Convergence of Margins
Cynthia Rudin, Ingrid Daubechies, Robert E. Schapire

Journal of Machine Learning Research 5 (2004) 801-818 Submitted 12/03; Published 7/04

Feature Discovery in Non-Metric Pairwise Data

Julian Laub JULIAN.LAUB@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekulestrasse 7
12489 Berlin, Germany

Klaus-Robert Müller KLAUS@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekulestrasse 7
12489 Berlin, Germany, and
University of Potsdam, Department of Computer Science
August-Bebel-Strasse 89
14482 Potsdam, Germany

Editor: Isabelle Guyon

Abstract

Pairwise proximity data, given as similarity or dissimilarity matrix, can violate metricity. This occurs either due to noise, fallible estimates, or due to intrinsic non-metric features such as they arise from human judgments. So far the problem of non-metric pairwise data has been tackled by essentially omitting the negative eigenvalues or shifting the spectrum of the associated (pseudo-)covariance matrix for a subsequent embedding. However, little attention has been paid to the negative part of the spectrum itself. In particular no answer was given to whether the directions associated to the negative eigenvalues would at all code variance other than noise related. We show by a simple, *exploratory* analysis that the negative eigenvalues *can* code for relevant structure in the data, thus leading to the discovery of new features, which were lost by conventional data analysis techniques. The information hidden in the negative eigenvalue part of the spectrum is illustrated and discussed for three data sets, namely USPS handwritten digits, text-mining and data from cognitive psychology.

Keywords: Feature discovery, exploratory data analysis, embedding, non-metric, pairwise data, unsupervised learning

1. Introduction

A large class of data analysis algorithms is based on a vectorial representation of the data. However, for major fields such as bioinformatics (e.g. Altschul et al., 1997), image analysis (Hofmann et al., 1998; Jacobs et al., 2000) or cognitive psychology (Gati and Tversky, 1982; Goldstone et al., 1991), the data is not available as points lying in some vector space but solely arises as scores of pairwise comparisons, typically measuring similarity or dissimilarity between the data points. A global overview of pairwise proximity data can be found in Everitt and Rabe-Hesketh (1997). Table 1 gives a simple instance of pairwise data obtained from human similarity judgments of Morse code (Everitt and Rabe-Hesketh, 1997).

	1	2	3	4	5
1	84	63	13	8	10
2	62	89	54	20	5
3	18	64	86	31	23
4	5	26	44	89	42
5	14	10	30	69	90

Table 1: Test subjects are asked to judge the pairwise similarity of auditory Morse code (long and short tones). Here we show the similarity matrix for the first five digits. The entries correspond to the percentage of a large number of observers who responded 'same' to the row signal followed by the column signal. (Excerpt from Table 1.3 given in Everitt and Rabe-Hesketh (1997)). Note that this proximity matrix is *asymmetric*.

These pairwise proximities, or "data points", are in no natural way related to the common viewpoint of objects lying in some "well behaved" space like a vector space which always is—albeit possibly high dimensional—a very restrictive structure.

There is a class of algorithms specifically designed for pairwise data, e.g. KNN, pairwise k-means (Duda et al., 2001). Otherwise the pairwise data is first "embedded" into a vector space in order to make it available for the numerous algorithms based on vectorial input (Cox and Cox, 2001). Pairwise data satisfying restrictive conditions can be embedded distortionless with respect to metricity into a Euclidean space (Roth et al., 2003b).

Often pairwise data is non-metric and the dissimilarity matrix does not satisfy the mathematical requirements of a metric function. Metric violations preclude the use of well established machine learning methods, which typically have been formulated for metric data only, and more precisely, for vectorial data, thus limiting the interest of raw pairwise data. Non-metric pairwise data can not be embedded distortionless into a (Euclidean) vector space. So, in general, embedding into a Euclidean space (and often subsequent dimension reduction) amounts to distorting pairwise data to enforce Euclidean metricity. This procedure is exemplified by Multi Dimensional Scaling (Cox and Cox, 2001). Other popular methods are e.g. *Locally Linear Embedding* (Roweis and Saul, 2000) and *Isomap* (Tenenbaum et al., 2000).

However, little is known about the real information loss incurred by enforcing metricity, when non-metric data is forcefully embedded into a vector space on the assumption that non-metricity be a mere artifact of noise. This assumption certainly holds for many cases, especially when the pairwise comparison is the output of some algorithm tuned to be metric but relying on some random initialization. It does not hold for pairwise data which is inherently non-metric, e.g. for human similarity judgments, summarizing geometrical (metric) and categorial thinking (possibly non-metric).

Technically, non-metricity translates into indefinite similarity matrices, also called "pseudo-covariance" matrices (Torgerson, 1958), a fact, which imposes severe constraints on the data analysis procedures. Typical approaches to tackle these problems involve omitting altogether the negative eigenvalues like in classical scaling (Young and Householder, 1938) or shifting the spectrum (Roth et al., 2003a) for subsequent (kernel-)PCA (Schölkopf et al., 1998).

The central and so far unanswered question is therefore: *Does the negative part of the spectrum of a similarity matrix code anything useful other than noise?*

This work will contribute by showing that the negative eigenvalues *can* code for relevant structure in the data. The *exploratory* technique outlined below can lead to the discovery of *systematically new* features, which were so far lost or have gone unnoticed by existing algorithms. This is discussed for three illustrative examples after a brief theoretic discussion of the issue of negative eigenvalues and a simple explanation of *how* negative spectra can code specific information.

2. Embedding of Non-Metric Data

In this section we will describe the issue of embedding pairwise proximity data into a vector space. Embedding pairwise data corresponds to finding points in a vector space, such that their mutual distance is as close as possible to the initial dissimilarity matrix with respect to some cost function. Embedding yields points in a vector space, thus making the data available to the numerous machine learning techniques which require vectorial input. Embedding also allows visualization after dimension reduction.

Let $D = (D_{ij})$ be an $n \times n$ dissimilarity matrix. We want to find n vectors x_i in a p-dimensional vector space such that the distance between x_i and x_j is close to the dissimilarity D_{ij} with respect to some cost function measuring the distortion incurred by the embedding procedure.

Let X be the matrix whose column are given by the vectors x_i. The matrix defined by $\frac{1}{n}XX^\top$ is called the *covariance* matrix and is positive semi-definite, i.e., all its eigenvalues λ_i are positive or zero ($\lambda_i \geq 0$). The covariance matrix plays an important role in spectral methods like (kernel-)PCA. The directions corresponding to the leading eigenvalues describe the directions which capture large variance in the data. Thus we expect to find interesting features there.

2.1 Mathematical Statement of the Embedding Problem

We will briefly state the mathematical formulation of the embedding problem and give the necessary and sufficient condition for a Euclidean embedding.

A dissimilarity matrix D will be called *metric* if there exists a metric function d such that $D_{ij} = d(\cdot,\cdot)$. In other words, D is positive and symmetric, its elements are 0 if and only if they are on the diagonal,[1] and they satisfy the triangle inequality. $D = (D_{ij})$ will be called squared-Euclidean if the metric function d derives from the Euclidean norm l_2.

Let $C = -\frac{1}{2}QDQ$ where $Q = I - \frac{1}{n}ee'$. Q is the projection matrix onto the orthogonal complement of $e = (1,1,\ldots 1)^\top$. The operation $D \rightarrow QDQ$ corresponds to the *centralizing* operation. The meaning of C will become clear subsequently.

We have the following important theorem (Torgerson, 1958; Young and Householder, 1938):

Theorem 1 *D is squared-Euclidean iff C is positive semi-definite.*

In other words, the pairwise dissimilarities given by D can be embedded into a Euclidean vector space if and only if the associated matrix C is positive semi-definite.

2.2 Embedding when D is Squared-Euclidean

When D is squared-Euclidean, C is semi-definite positive and can readily be interpreted as covariance matrix (via simple algebra). The embedded vectors can be recovered by usual kernel-PCA (Schölkopf et al., 1998; Cox and Cox, 2001):

1. We reasonably suppose that there are no two identical data points with different labels in the data set.

$$D \xrightarrow{C=-1/2QDQ} C \text{ with } n \text{ positive eigenvalues}$$
$$C \xrightarrow{\text{spectral decomposition}} V\Lambda V^\top$$
$$X_K = |\Lambda_K|^{1/2} V_K^\top,$$

where $V = (v_1, \ldots, v_n)$ with eigenvectors v_i's and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$. K is the subspace of chosen directions v_i. The columns of X_K contain the vectors x_i in p-dimensional subspace K, where V_K is the column-matrix of the selected eigenvectors and Λ_K the diagonal matrix of the corresponding eigenvectors.

If $K = \{v_1 \ldots v_p\}$ the distances between these vectors differ the least from the distances D with respect to the quadratic approximation error. For $p = n-1$ the mutual distances coincide with D, i.e. $D_{ij} = ||x_i - x_j||^2$. In other words: *there is a direct algebraic transformation between D and the set of x_i's.*

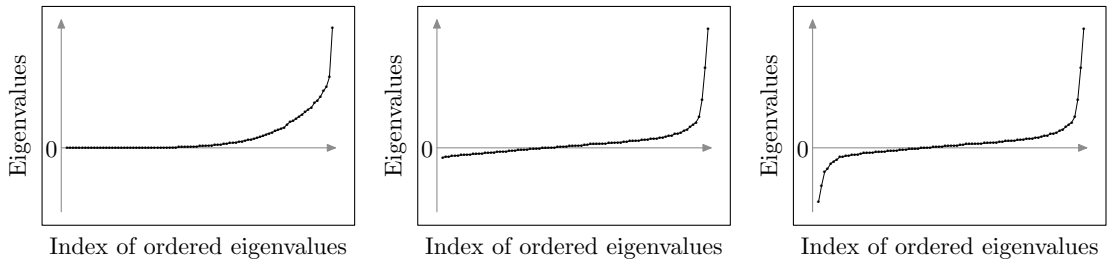


Figure 1: Examples of spectra of C for a squared-Euclidean dissimilarity (left) and non squared-Euclidean dissimilarity matrices. The eigenvalues are plotted against rank order.

2.3 Embedding for General D's

For non squared-Euclidean dissimilarity matrices D the associated C is not positive semi-definite and is not a covariance matrix. In this case we will call it *pseudo-covariance* matrix. Figure 1 shows an instance of a spectrum associated to a positive semi-definite C (left) and two instances of negative spectra: in the middle, a spectrum associated to a pairwise dissimilarity which essentially is metric but corrupted by noise which translates into a spectrum with only a few negative eigenvalues of small magnitude. On the right, a spectrum with negative eigenvalues large in magnitude associated to intrinsic non-metricity.

In order to study the possible loss incurred by omitting the negative part of the spectrum we propose the following simple algorithm, which allows to specifically visualize the information coded by the negative eigenvalues.

Algorithm. Start with some symmetric dissimilarity D or similarity S. If non-symmetric cases the pairwise proximity matrix must first be symmetrized. Furthermore, when the proximity data are similarities, a problem specific dissimilarity is computed. D typically is related to S via, e.g., $D_{ij} = 1 - S_{ij}$ or $D_{ij} = S_{ii} + S_{jj} - 2S_{ij}$. Recall that $Q = I - \frac{1}{n}ee'$ with $e = (1, 1, \ldots 1)^\top$. C denotes the

pseudo-covariance matrix.

$$
\begin{aligned}
&D \xrightarrow{C=-1/2QDQ} C \text{ with } p \text{ positive and } q \text{ negative eigenvalues} \\
&C \xrightarrow{\text{spectral decomposition}} V\Lambda V^\top = V|\Lambda|^{\frac{1}{2}} M |\Lambda|^{\frac{1}{2}} V^\top \\
&X_K = |\Lambda_K|^{1/2} V_K^\top,
\end{aligned}
$$

where M is the block matrix consisting of the blocks $I_{p\times p}$, $-I_{q\times q}$ and $0_{k\times k}$ (with $k = n-p-q$).

The columns of X_K contain the vectors x_i in the p-dimensional subspace K. At this point K can be very general. However, as for PCA, we will find it sensible to choose a few leading eigendirections *which can also include eigendirections associated to the negative part of the spectrum*.

Visualization. Retaining only the first two coordinates ($K = \{v_1, v_2\}$) of the obtained vectors corresponds to a projection onto the first two leading eigendirections. Retaining the last two ($K = \{v_{n-1}, v_n\}$) is a projection onto the last two eigendirections: *This corresponds to a projection onto directions related to the metric violations of C*

This simple algorithm[2] is very akin to the well known PCA algorithm except that it does not require the spectrum to be positive.

To finish this short overview about embedding pairwise data, it is important to stress the fact that in general metricity is not enough for pairwise data to be loss-free embeddable into a Euclidean vector space. Pairwise data may be metric, yet have an associated spectrum with negative eigenvalues. The interesting case, however, is given for the Euclidean metric since Theorem 1 establishes a very simple relationship between the requirements on the dissimilarity matrix and its loss-free embeddability into a Euclidean vector space.

2.4 Issue of Information Loss

Classical approaches to the embedding into a Euclidean vector space usually involve techniques like multi-dimensional scaling (Cox and Cox, 2001). In its simplest version, classical scaling, MDS proceeds as the algorithm in Section 2.1. However $\Lambda_K^{1/2}$ is only defined for $K \subset \{v_1 \dots v_p\}$ with $p \leq t$ where t is the number of positives eigenvalues. The requirement $p \leq t$ leads to a cut-off of the negative eigenvalues. Another variant of MDS is called *non-metric* MDS and treats ordinal-scale data, where the projections only try to preserve the rank order between the distances, not their absolute value (Kruskal, 1964; Shepard, 1962). It is important to notice here that in our work non-metricity refers to the violations of metric requirements and the subsequent impossibility of a loss-free embedding into a Euclidean vector space. Non-metric MDS does not discover the information coded specifically by metric violations.

Recently *Constant Shift Embedding* was introduced which guarantees distortionless embedding of non-metric pairwise data w.r.t. cluster assignment in the case of a shift invariant cluster cost function (Roth et al., 2003b,a). However, in practical applications, the need for dimension reduction to speed up optimization and robustify solutions, effectively results in retaining only the leading eigendirections and cutting off large parts of the spectrum. For other cases than noise corrupted non-metric pairwise data (Figure 1, middle) it is an open question whether the removal of negative eigenvalues leads to an information loss.

2. A Matlab implementation can be found under `http://ida.first.fraunhofer.de/~jlaub/`.

The above methods, unlike ours, do not permit to specifically study the information coded by non-metricity.

3. Interpretation and Modeling of Negative Spectra

In this section we will discuss the issue of information loss raised by the preceding considerations. We will first show by simple from-scratch constructions how negative spectra can occur. Further understanding will be gained by interpretation of the negative eigenspaces. In particular, a model is presented that can explain negative spectra in the case of human similarity judgments in cognitive psychology. A simple toy illustration will conclude this section.

3.1 Constructing Negative Spectra

Let us first introduce two simple constructions of non-metric pairwise data sets whose non-metric part codes for specific information. These constructions typically come about in situations involving penalization and/or competition of dissimilarity measures by subtraction or division:

$$D_{ij} = (D_1 - D_2)_{ij} \quad \text{or} \quad D_{ij} = \frac{(D_1)_{ij}}{(D_2)_{ij}}, \tag{1}$$

with the assumption that $(D_2)_{ij} \neq 0 \; \forall i, j = 1, 2 \ldots n$ in the latter case. See Figure 2 for a schematic illustration. The structure of the penalized cells is reflected in non-metricity. Such similarity scores occur in various image matching algorithms or in text mining via e.g. the *min-max* dis/similarity (Banerjee and Ghosh, 2002; Dagan et al., 1995), but also in alignment algorithms from e.g. bioinformatics.

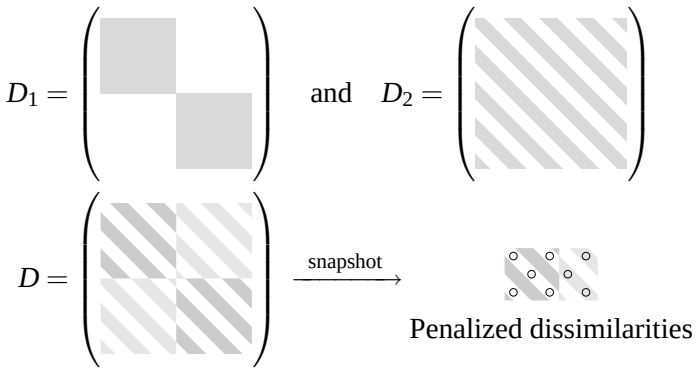


Figure 2: Principle of penalization: the penalized cells can form a structure on their own, which is reflected in non-metricity. The snapshot shows the alternate structure of the penalized entries (small circles).

3.2 Interpreting Negative Eigenspaces

For a positive semi-definite C the projections along the leading eigendirections can readily be interpreted as projections along the axis of high variances of the data. For pseudo-covariance matrices

this still holds up to a scaling factor when shifting the spectrum so as to assure positive semi-definiteness.

For projections onto the negative eigendirections the interpretation is not so straightforward since there is no clear intuition on what "negative variance" represents. However the above presented algorithm relies on a pseudo-Euclidean-style decomposition of the embedding space. The pseudo-Euclidean space effectively amounts to two Euclidean spaces one of which has a positive semi-definite inner product and the other a negative semi-definite inner product. An interesting interpretation of the distances in a squared-Euclidean space is that they can be looked at as a difference of squared-Euclidean distances from the "positive" and the "negative" space, by the decomposition $\mathbb{R}^{(p,q)} = \mathbb{R}^p + i\mathbb{R}^q$, so that $D_{ij} = D_{ij}^{(\mathbb{R}^p)} - D_{ij}^{(\mathbb{R}^q)}$, where the D_{ij} are squared-Euclidean. This is the rationale behind the first construction of a non-metric D via $D_{ij} = (D_1 - D_2)_{ij}$.

The power of this decomposition resides in the fact that the negative eigenvalues now admit the natural interpretation of variances of the data projected onto directions in $\mathbb{R}^q$. Thus the variance along v_n is $\sqrt{|\lambda_n|}$, the variance along v_{n-1} is $\sqrt{|\lambda_{n-1}|}$, etc. (c.f. Pękalska et al., 2001).

3.3 Modeling Negative Spectra

While the dissimilarity matrix constructions obtained from Equation 1 can account for a class of non-metric pairwise data from domains such as image matching, text-mining or bioinformatics, where penalization models underlie the computed similarities, they cannot necessarily appropriately model the generic case where a pairwise dissimilarity matrix is given from an experiment, say in cognitive psychology. The simple model for non-metric pairwise data introduced in the following is inspired by approaches in cognitive psychology to explain human similarity judgments (Tenenbaum, 1996). Let $\{f_1, f_2, \dots f_n\}$ be a basis. A given data point x_i can be decomposed in this basis as $x_i = \sum_{k=1}^n \alpha_k^{(i)} f_k$. The squared l_2 distance between x_i and x_j therefore reads: $d_{ij} = ||x_i - x_j||^2 = \left|\left| \sum_{k=1}^n \left(\alpha_k^{(i)} - \alpha_k^{(j)}\right) f_k \right|\right|^2$. However this assumes constant feature-perception, i.e. a constant mental image with respect to different tasks. In the realm of human perception this is often not the case, as illustrated by the following well known visual "traps" (Figure 3). Our perception has several ways to interpret the figures which can give rise to asymmetry (Thomas and Mareschal, 1997) by a different weighting of the perceived dissimilarities. It is important to notice here that for human similarity judgments, one can hardly speak of artifact or erroneous judgments with respect to a Euclidean norm. The latter seems rather exceptional in these cases.

A possible way to model different interpretations of a given geometric object is to introduce states $\{\omega^{(1)}, \omega^{(2)} \dots \omega^{(d)}\}$, $\omega^{(l)} \in \mathbb{R}^n$ for $l = 1, 2, \dots d$, affecting the features. The similarity judgment between objects then depends on the perceptual state (weight) the observer is in. Assuming that the person is in state $\omega^{(l)}$ the distance becomes:

$$d_{ij} = ||x_i - x_j||^2 = \left|\left| \sum_{k=1}^{n} \left(\alpha_k^{(i)} - \alpha_k^{(j)}\right) \omega_k^{(l)} f_k \right|\right|^2. \tag{2}$$

With no further restriction this model yields non metric distance matrices.

In the worst case l is random, but usually perception-switches can be modeled and l becomes some function of (i, j). For random l, non-metricity is an artifact of sample size, since when averaging the d's over p observers the mean dissimilarity is asymptotically metric in p ($\langle d \rangle \to$ metric as $p \to \infty$): the mean ω becomes constant for all i, j equal to the expectation of its distribution.

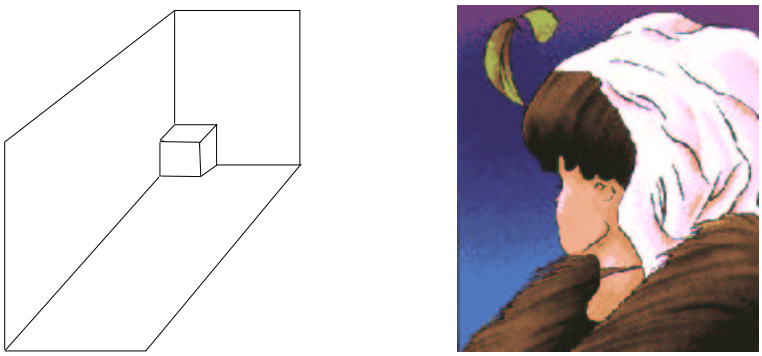


Figure 3: Left: What do you see? A small cube in the corner of a room or a large cube with a cubic hole or a small cube sticking with one corner on a large one? Right: What do you see? A young lady or an old woman? If you were to compare this picture to a large set of images of young ladies or old women, the (unwilling) perception switch could induce large individual weights on the similarity.

On the other hand, if we suppose that the function l of (i, j) does not vary much between observers, then the averaging does not flatten out the non-metric structure induced by the systematic perception-switch.

3.4 Illustration (Proof of Principle)

The importance of the information coded by the negative eigenvalues is exemplified by the following simple example: consider n objects, labeled $1, 2, \dots n$, presenting two salient features. Suppose that they cluster into $\{1, \dots \frac{n}{2}\}$ and $\{\frac{n}{2}+1, \dots n\}$ according to the first feature, and into $\{1, 3, 5, \dots n-1\}$ and $\{2, 4, 6, \dots n\}$ according to the second. Let D_1 and D_2 be the dissimilarity matrices corresponding to feature 1 and 2 respectively. Save very pathological cases the spectra associated to the D obtained by subtraction or division of D_1 and D_2 contain steeply falling negative eigenvalues. Furthermore the projection onto the first two eigenvalues exhibits a clear distinction w.r.t. feature 1 whereas the projection onto the last two eigenvalues exhibits a clear distinction w.r.t. feature 2.

Let e.g. $n = 8$. Two artificial dissimilarity matrices D_1 and D_2 were constructed to reflect the above surmise about the underlying structure:

$$D_1 = \begin{pmatrix} 0.00 & 2.36 & 2.59 & 1.78 & 4.74 & 4.82 & 4.98 & 4.72 \\ 2.36 & 0.00 & 2.39 & 1.60 & 4.98 & 5.06 & 5.22 & 4.96 \\ 2.59 & 2.39 & 0.00 & 2.09 & 5.29 & 5.37 & 5.53 & 5.27 \\ 1.78 & 1.60 & 2.09 & 0.00 & 5.08 & 5.16 & 5.32 & 5.06 \\ 4.74 & 4.98 & 5.29 & 5.08 & 0.00 & 1.20 & 1.82 & 1.62 \\ 4.82 & 5.06 & 5.37 & 5.16 & 1.20 & 0.00 & 2.98 & 1.78 \\ 4.98 & 5.22 & 5.53 & 5.32 & 1.82 & 2.98 & 0.00 & 2.02 \\ 4.72 & 4.96 & 5.27 & 5.06 & 1.62 & 1.78 & 2.02 & 0.00 \end{pmatrix} \text{ and } D_2 = \begin{pmatrix} 0.00 & 4.15 & 2.03 & 4.14 & 1.26 & 4.33 & 0.690 & 4.85 \\ 4.15 & 0.00 & 4.70 & 0.570 & 4.37 & 1.82 & 4.24 & 2.02 \\ 2.03 & 4.70 & 0.00 & 4.69 & 1.85 & 4.88 & 1.68 & 5.40 \\ 4.14 & 0.570 & 4.69 & 0.00 & 4.36 & 1.83 & 4.23 & 2.67 \\ 1.26 & 4.37 & 1.85 & 4.36 & 0.00 & 4.55 & 0.730 & 5.07 \\ 4.33 & 1.82 & 4.88 & 1.83 & 4.55 & 0.00 & 4.42 & 2.14 \\ 0.690 & 4.24 & 1.68 & 4.23 & 0.730 & 4.42 & 0.00 & 4.94 \\ 4.85 & 2.02 & 5.40 & 2.67 & 5.07 & 2.14 & 4.94 & 0.00 \end{pmatrix}.$$

Figure 4 shows the result obtained by Algorithm 2.3. The spectrum associated to $D = D_1 - D_2$ is non positive. The information contained in the positive and the negative part is recovered: We see that the information represented in the first two eigendirections is related to the variance due to the cluster structure $\{1, \dots 4\}$ and $\{5, \dots 8\}$ whereas the information represented in the last two eigendirection relates to the cluster structure $\{1, 3, \dots 7\}$ and $\{2, 4, \dots 8\}$. *This last information*

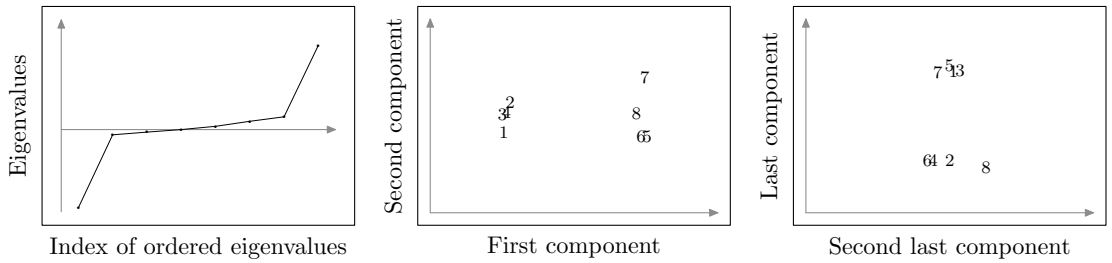


Figure 4: Sorted spectrum associated to the non-metric E (right), Projection onto the two leading positive eigendirection and projection onto the two leading negative eigendirections (right).

would have been lost by usual methods relying on high variance and thus neglecting the negative eigenvalues.

4. Summary

We summarize the procedure and the rationale behind it (see *schematic* diagram Figure 5).

Consider the following illustrative setting: we have apples of different sizes and two colors. There are two salient features: size (geometric) and color (categorial). These apples are pairwise compared, either by a computer algorithm, a human test subject or any other mechanism. This comparison yields a dissimilarity matrix D or a similarity matrix S. In the latter case a problem specific dissimilarity matrix D is obtained from S.

From D we compute the centralized (pseudo-)covariance matrix C and its spectrum. C is positive semi-definite if and only if D is squared-Euclidean. For generic D this is not the case and the usual techniques fail to take this into account.

We project the data onto the first two leading eigenvectors explaining the variance associated to the first feature (size). Second, we project the data onto the last two eigenvectors accounting for the variance of the second feature (color). This last step is done by an embedding into the pseudo-Euclidean space.

The second feature is lost by any method relying exclusively on high variance, that is, the majority of machine learning techniques. We propose the exploration of the negative eigenspectrum for *feature discovery*.

5. Further Illustrative Applications

To go beyond toy examples showing that non-metricity can code for interesting features, we will now illustrate our feature discovery technique by three applications from real-world domains, namely image matching, text mining and cognitive psychology.

5.1 USPS Handwritten Digits

A similarity matrix is computed from binary image matching on the digits 0 and 7 of the USPS data set. Digits 0 and 7 have been chosen since they exhibit clear geometric differences. All images have

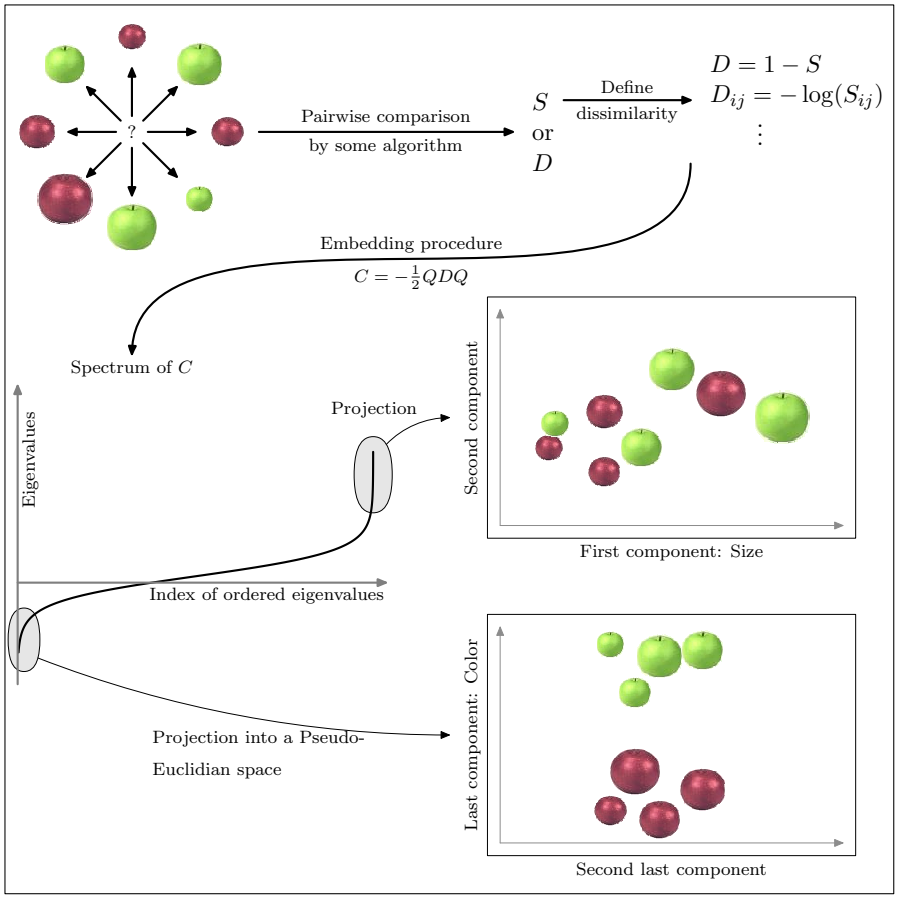


Figure 5: Summarizing diagram.

been sorted according to decreasing sum of pixel value (1 to 256) thus separating the bold digits from the light ones. A total of 1844 samples have been retained. The images have been normalized and discretized to have binary pixel values 0 and 1.

Binary image matching. Let r and s denote the labels of two images and S_{rs} the score rating mutual similarity. In the case of binary images, S_{rs} is a function of a, b, c and d, where a counts the number of variables where both objects s and r score 1, b the number where r scores 1 and s scores 0, c the number of variables where r scores 0 and s scores 1 and d the number of where both objects score 0. The counting variables a, b, c and d allow to define a variety of similarity scores S_{rs} (see Cox and Cox, 2001). We will be interested in the *Simpson* score, defined by

$$S_{rs} = \frac{a}{\min(a+b, a+c)}.$$

It exhibits a strongly falling negative spectrum, corresponding to highly non-metric data. Projection onto the eigenvectors associated to the first leading eigenvalues and projection onto the eigenvectors associated to the last eigenvalues yield results different in nature.

In each case there is a clear interpretation of the variance according to salient features: (i) Figure 6 shows that the variance in the "positive" eigenvectors corresponds to the geometrical dis-

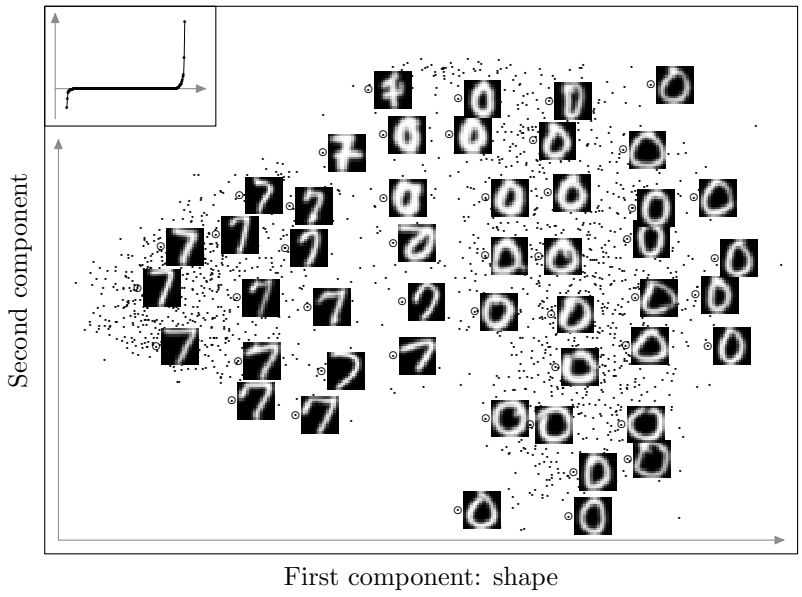


Figure 6: Projection onto the first two positive eigendirections. The explained variance is associated to the geometric shape.

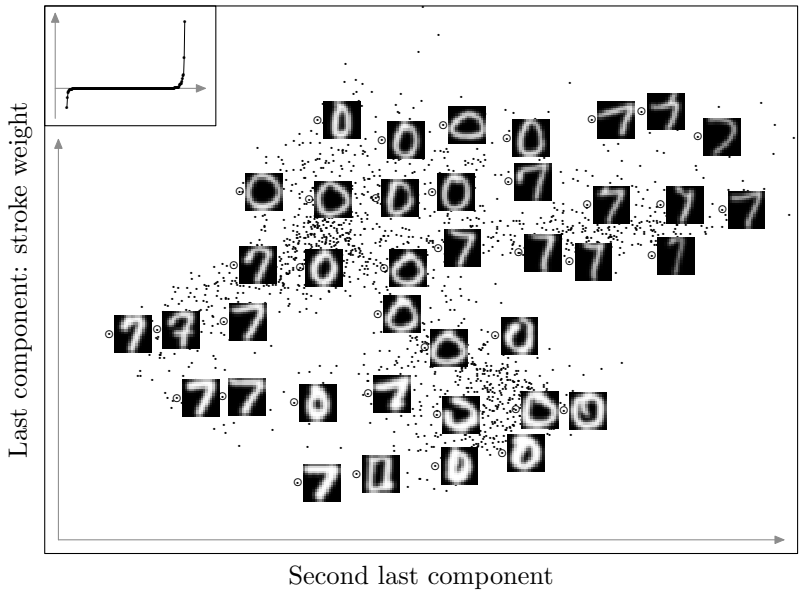


Figure 7: Projection onto the last two negative eigendirections. The explained variance is associated to the stroke weight.

tinction between the shapes of the 0's and the 7's; (ii) Figure 7 on the other hand shows that in the "negative" eigenvectors the variance is associated to the feature of stroke weight. *This inter-*

esting feature would have been lost if we had embedded the data by conventional methods thereby discarding the negative part of the spectrum.

5.2 Text-Mining

We are interested in the semantic structure of nouns and adjectives from two topically unrelated sources, namely Grimm's Fairy Tales (Gutenberg) and popular science articles about space exploration (NASA). Both sources contributed 60 documents containing roughly between 500 and 1500 words each. A subset of 120 nouns and adjectives has been selected, containing both very specific and very general terms out of both data sources.

Similarity measure for words. From a set of p documents and a choice of n keywords we can construct a contingency table, by simply indicating whether word i ($1 \leq i \leq n$) appears in document k ($1 \leq k \leq p$) or not. This yields a $p \times n$ boolean matrix.

We will take the *Keyword Semantic Proximity* as similarity measure (Rocha, 2001; Rocha and Bollen, 2001), which expresses that two words are similar if they often appear together in a document. This similarity is penalized if they individually spread over a large number of documents:

$$s_{ij} = \frac{\#\{\text{documents where word } i \textit{ and } \text{word } j \text{ appear}\}}{\#\{\text{documents where word } i \textit{ or } \text{word } j \text{ appear}\}}.$$

From this similarity measure, we obtain a dissimilarity matrix via, e.g. $d_{ij} = -\log(s_{ij})$. In Rocha (2001) the author uses $d_{ij} = 1/s_{ij} - 1$ which is another possible choice. In either case, the resulting dissimilarity matrix d is not squared-Euclidean such that the associated (pseudo-)covariance matrix exhibits strong negative eigenvalues (see inset in Figure 8).

The data is projected on the first two leading eigenvectors (Figure 8). On the far left we find the words stemming from the popular science articles (e.g. "nuclear", "computer", "physics" etc.) whereas on the far right we have those from Grimm's Fairy Tales (e.g. "castle", "queen", "ravens" etc.). The captured variance can be interpreted as the semantic context of the words.

Projection onto the last two eigendirections yields a distribution over a new interesting feature (Figure 9). We notice that in the upper half we find words of high specificity of either of the sources (e.g. "astronauts", "wolf", "witch" etc.). In the lower half we see an accumulation of words with general, unspecific, meaning, expected to be found in a large variety of documents (e.g. "day", "world", "thing" etc.). Thus to our understanding the variance associated to the last eigendirection again corresponds to the *specificity* of the words (relative to the data source). *This feature would have gone unnoticed by algorithms not specifically taking into account the negative eigenvalues.*

5.3 Cognitive Psychology

We finally present an example from human similarity judgments in cognitive psychology. This will also allow us to illustrate the model presented in Section 3.3.

The pairwise dissimilarity data is obtained from Gati and Tversky (1982). The stimuli tested consist of 16 images of flowers having leaves of varying elongation and stems of increasing size (Figure 10). These two stimuli were presented to a group of thirty undergraduate students from the Hebrew University who, individually, evaluated the mutual dissimilarity of the flowers on a 20-point scale (see Gati and Tversky, 1982).

We have processed the data according to Algorithm 2.3. In the positive eigendirections we obtain a very good reconstruction of the two geometric features, namely the elongation of the leaves

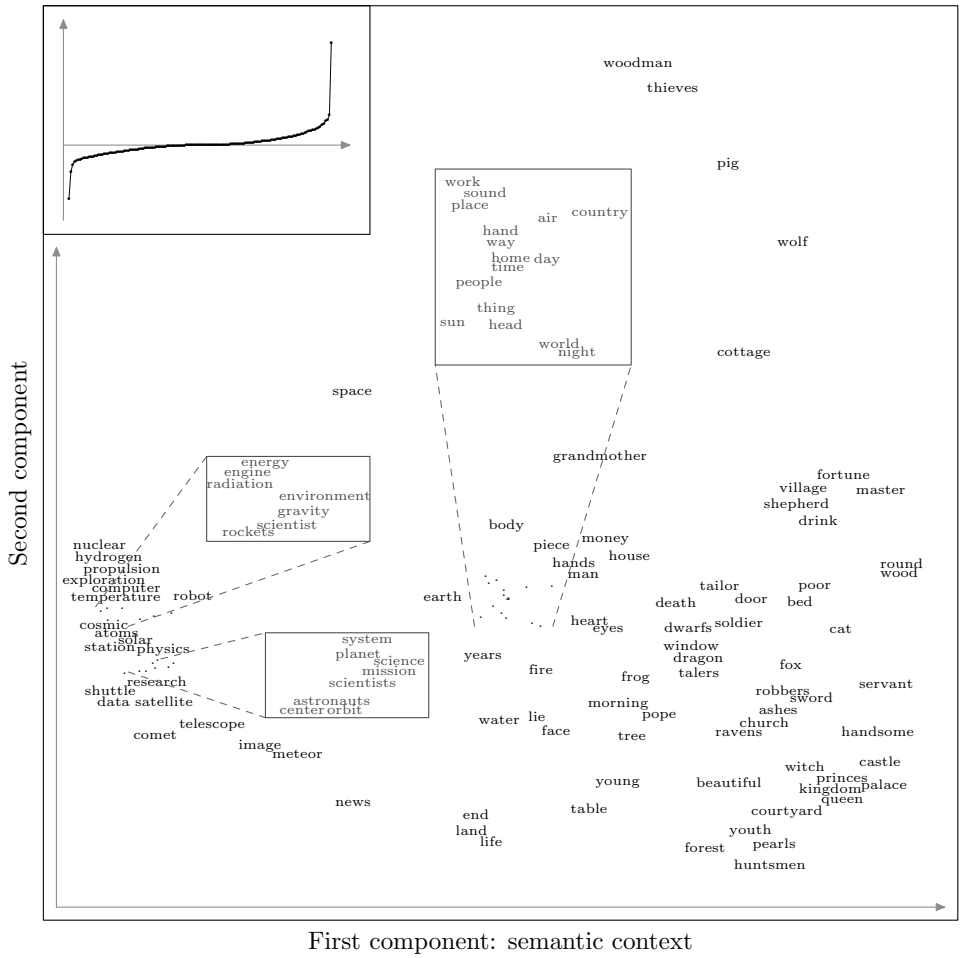


Figure 8: Projection onto the first two eigendirections.

and the size of the stem: see Figure 11, middle. There seems to be no tendency to favor one over the other. The first component explains the variance in leaf elongation (horizontal axis), the second the variance of the stem size (vertical axis).

Interestingly, the projection onto the last two negative eigendirections exhibits further structure, as shown by Figure 11, right. The interpretation, however, is not so straightforward. Two clusters loosely form, separated by the last eigendirection (vertical axis). They are $\{1,2,5,6,11,12,15,16\}$ and $\{3,4,7,8,9,10,13,14\}$. A possible feature could be the oddness of a plant, such that the first cluster contains the odd plants, and the second the "normal" ones, since one could expect plants with small leaves to be of small size and plants with large leaves to be of greater size. The odds here are the small plants with large leaves and the large plants with small leaves. This would correspond to categorial perception while judging similarity.

Features related to the concept of normality, or expectation, are not uncommon in cognitive psychology. In Navarro and Lee (2002) features like the normality or usuality of faces are discussed in the context of the Modified Contrast Model, along with certainly not easily graspable features like relationships in parenthood. While the authors focus on common and distinctive features and

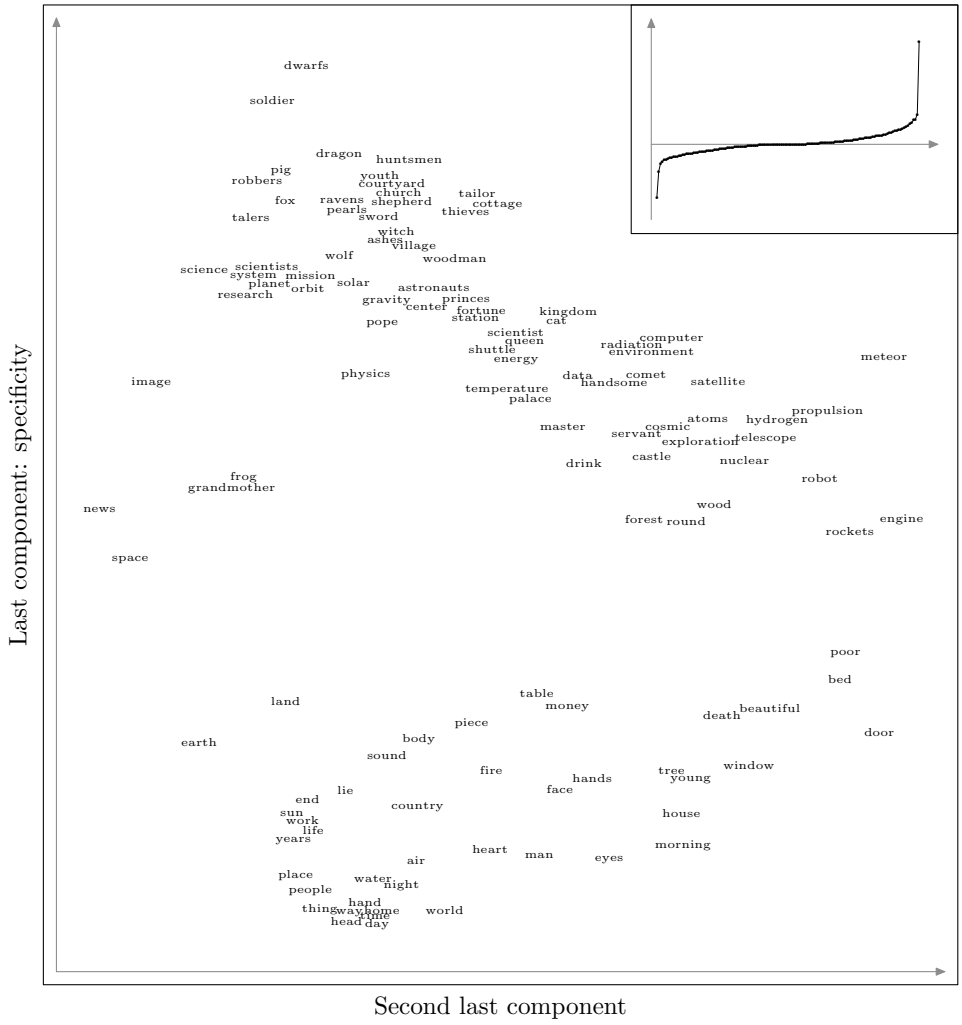


Figure 9: Projection onto the last two negative eigendirections.

distinguish between conceptual and perceptual features, the interpretation of the discovered features remains as a second independent step in data analysis.

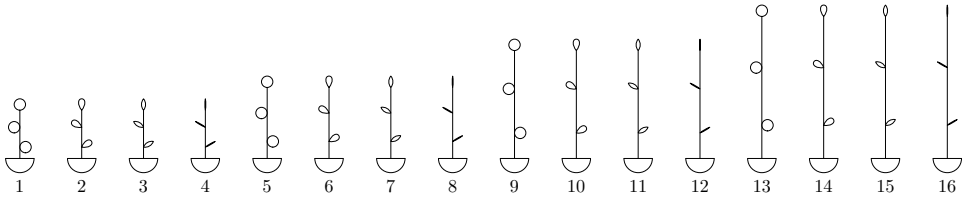


Figure 10: Images of the flowerpots as presented to the test person.

We explain the flowerpot experiment according to the model presented in Section 3.3, starting from a uniform distribution of 16 points in three dimensions and choosing the feature vectors f_k, $k = 1,2,3$ to be the unit vectors $e_1 = (1,0,0)$ etc.

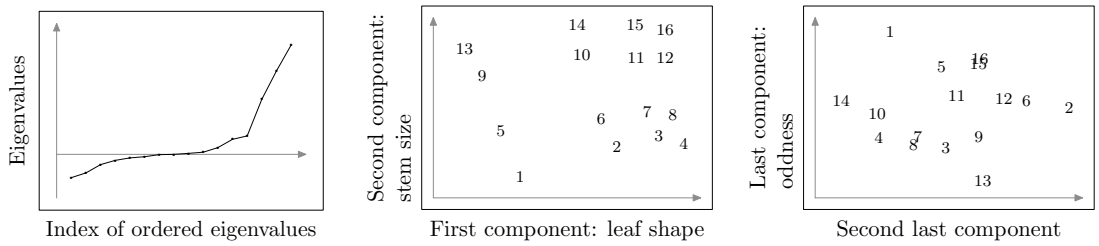


Figure 11: Left: Sorted eigenvalues. Middle: Projection onto the leading two positive eigenvalues. Right: projection onto the last two eigendirections.

The states are obtained by fitting the d defined in Equation 2 heuristically to the experimental dissimilarity by minimization of the difference of the means over all matrix elements.

We obtain a good model fit for six states $\{(8.3,0,0),\ (0,3.5,0),\ (4.7,4.7,4.7),\ (6.4,6.4,0),\ (0,3.4,3.4),\ (3.1,0,3.1)\}$. See Figure 12.

In other words, following the semantics of the model presented, one can explain the results of the obtained dissimilarities by six perceptual states of the observer; i.e. the weight vectors model the bias in perception. These seem to outnumber the actually observed features (in the two-dimensional representations) which are three in number (the two geometric features in the positives and the categorial one in the negatives). However, we must keep in mind that one may reduce the number of weights required to approximate d by a deeper knowledge of the initial feature presentation, including its dimensionality. We have taken a uniform distribution in three dimensions for lack of more precise knowledge.

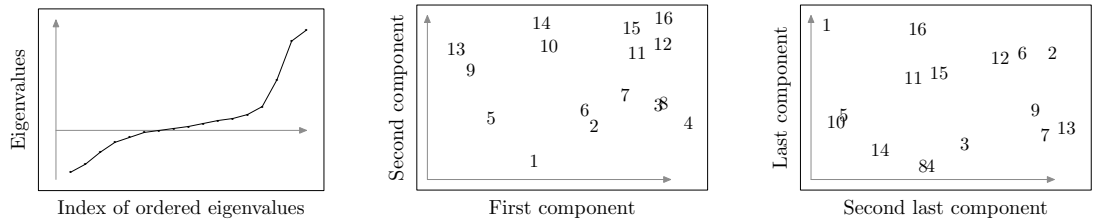


Figure 12: Prediction of flowerpot experiment.

6. Conclusion and Outlook

This work studies the potential of relevant information being coded specifically by the non-metric part of the spectrum of a pseudo-covariance matrix. It has been shown that non-metricity *can* indeed code for features relevant to a better understanding of the data set. The proposed algorithm effectively overcomes the drawback of most variance based algorithms which take only into account the variance of the leading eigendirections. Model.illustrations provide a simple intuition and explanation of the phenomena. Note, however, that spectra like Figure 1 (right) are only potentially—not necessarily—containing interesting information in their negative part, some fancy noise process might also be the cause of such a structure.

Concluding, it is an important step in unsupervised data analysis to find out whether the negative part of the spectrum codes for interesting variance. The present technique can be employed as a general exploratory feature discovery tool. Application fields range from cognitive psychology, marketing, biology, engineering and bioinformatics, where in principle new structure awaits its discovery.

A further interesting direction is to go beyond visualization toward automated structure learning. Investigations to overcome low-dimensional feature discovery based on visualization will focus on, e.g., stability analysis (Roth et al., 2002; Meinecke et al., 2002) of various projections onto the possibly negative eigenspace in order to assess quantitatively relevant structure and to rule out noise related, erroneous, feature interpretations.

A major focus will concern the automated distinction of structure induced by intrinsic non-metricity from mere artifacts of some fancy noise process with the overall goal to provide automated learning and procedures that can optimally make use of the information coded by intrinsic non-metricity.

Acknowledgments

Special thanks go to Volker Roth for valuable discussions, as well as to Motoaki Kawanabe, Christin Schäfer, Sebastian Mika, Mikio Braun and Andreas Ziehe for thorough reading of the manuscript. This work is partially supported by DFG grants # MU 987/1-1 and # JA 379/13-2 as well as PASCAL Network of Excellence (EU # 506778).

A particular acknowledgment goes to the reviewers and the action editor who helped with many stringent and interesting remarks and suggestions improve this work.

References

S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

A. Banerjee and J. Ghosh. Clickstream clustering using weighted longest common subsequences. *In Proceedings of the Web Mining Workshop at the 1 st SIAM Conference on Data Mining, Chicago, April 2001.*, 2002.

T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 2001.

I. Dagan, S. Marcus, and S. Markovitch. Contextual word similarity and extimation from sparse data. *Computer Speech and Language*, (9(2)):123–152, 1995.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, second edition, 2001.

B. S. Everitt and S. Rabe-Hesketh. *The Analysis of Proximity Data*. Arnold, London, 1997.

I. Gati and A. Tversky. Representation of qualitative and quantitative dimensions. *Journal of Experimental Psychology: Human Perception and Performance*, 8(2):325–340, 1982.

R. L. Goldstone, D. L. Medin, and D. Gentner. Relational similarity and the nonindependence of features in similarity judgments. *Cognitive Psychology*, pages 222–262, 1991.

Project Gutenberg. `http://promo.net/pg/`.

T. Hofmann, J. Puzicha, and J. M. Buhmann. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8): 803–818, 1998.

D. W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):583–600, 2000.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29, 1964.

F. Meinecke, A. Ziehe, M. Kawanabe, and K.-R. Müller. A resampling approach to estimate the stability of one-dimensional or multidimensional independent components. *IEEE Transactions on Biomedical Engineering*, 49:1514–1525, 2002.

NASA. `http://science.nasa.gov/headlines/news_archive.htm`.

D. J. Navarro and M. D. Lee. Common and distinctive features in stimulus similarity: A modified version of the contrast model. *submitted*, 2002.

E. Pękalska, P. Paclík, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, (2):175–211, 2001.

L. M. Rocha. Talkmine: A soft computing approach to adaptive knowledge recommendation. *Soft Computing Agents: New Trends for Designing Autonomous Systems*, pages 89–116, 2001.

L. M. Rocha and J. Bollen. Biologically motivated distributed designs for adaptive knowledge management? *Design Principles for the Immune System and other Distributed Autonomous Systems*, pages 305–334, 2001.

V. Roth, T. Lange, M. Braun, and J. M. Buhmann. A resampling approach to cluster validation. *Statistics–COMPSTAT*, pages 123 – 128, 2002.

V. Roth, J. Laub, J. M. Buhmann, and K.-R. Müller. Going metric: Denoising pairwise data. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 817–824. MIT Press: Cambridge, MA, 2003a.

V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of non-metric proximity data. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25 (12):1540–1551, 2003b.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2000.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

R. N. Shepard. The analysis of proximities: multidimensional scaling with an unknown distance function. *Psychometrika*, 27, 1962.

J. B. Tenenbaum. Learning the structure of similarity. *Advances in Neural Information Processing Systems*, 1996.

J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2000.

M. S. C. Thomas and D. Mareschal. Connectionism and psychological notions of similarity. *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, 1997.

W. S. Torgerson. *Theory and Methods of Scaling*. John Wiley and Sons, New York, 1958.

G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22, 1938.

Journal of Machine Learning Research 5 (2004) 819–844 Submitted 1/04; Published 7/04

Probability Product Kernels

Tony Jebara JEBARA@CS.COLUMBIA.EDU
Risi Kondor RISI@CS.COLUMBIA.EDU
Andrew Howard AHOWARD@CS.COLUMBIA.EDU
Computer Science Department
Columbia University
New York, NY 10027, USA

Editors: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

The advantages of discriminative learning algorithms and kernel machines are combined with generative modeling using a novel kernel between distributions. In the probability product kernel, data points in the input space are mapped to distributions over the sample space and a general inner product is then evaluated as the integral of the product of pairs of distributions. The kernel is straightforward to evaluate for all exponential family models such as multinomials and Gaussians and yields interesting nonlinear kernels. Furthermore, the kernel is computable in closed form for latent distributions such as mixture models, hidden Markov models and linear dynamical systems. For intractable models, such as switching linear dynamical systems, structured mean-field approximations can be brought to bear on the kernel evaluation. For general distributions, even if an analytic expression for the kernel is not feasible, we show a straightforward sampling method to evaluate it. Thus, the kernel permits discriminative learning methods, including support vector machines, to exploit the properties, metrics and invariances of the generative models we infer from each datum. Experiments are shown using multinomial models for text, hidden Markov models for biological data sets and linear dynamical systems for time series data.

Keywords: Kernels, support vector machines, generative models, Hellinger divergence, Kullback-Leibler divergence, Bhattacharyya affinity, expected likelihood, exponential family, graphical models, latent models, hidden Markov models, dynamical systems, mean field

1. Introduction

Recent developments in machine learning, including the emergence of support vector machines, have rekindled interest in kernel methods (Vapnik, 1998; Hastie et al., 2001). Typically, kernels are designed and used by the practitioner to introduce useful nonlinearities, embed prior knowledge and handle unusual input spaces in a learning algorithm (Schölkopf and Smola, 2001). In this article, we consider kernels between distributions. Typically, kernels compute a generalized inner product between two input objects χ and χ' which is equivalent to apply a mapping function Φ to each object and then computing a dot product between $\Phi(\chi)$ and $\Phi(\chi')$ in a Hilbert space. We will instead consider the case where the mapping $\Phi(\chi)$ is a probability distribution $p(x|\chi)$, thus restricting the Hilbert space since the space of distributions is trivially embedded in the Hilbert space of functions. However, this restriction to positive normalized mappings is in fact not too limiting since general $\Phi(\chi)$ mappings and the nonlinear decision boundaries they generate can often be mimicked by a probabilistic mapping. However, a probabilistic mapping facilitates and elucidates

kernel design in general. It permits kernel design to leverage the large body of tools in statistical and generative modeling including Bayesian networks (Pearl, 1997). For instance, maximum likelihood or Bayesian estimates may be used to set certain aspects of the mapping to Hilbert space.

In this paper, we consider the mapping from datum x to a probability distribution $p(x|x)$ which has a straightforward inner product kernel $k(x,x') = \int p^\rho(x|x)p^\rho(x|x')dx$ (for a scalar ρ which we typically set to 1 or $1/2$). This kernel is merely an inner product between two distributions and, if these distributions are known directly, it corresponds to a linear classifier in the space of probability distributions. If the distributions themselves are not available and only observed data is given, we propose using either Bayesian or Frequentist estimators to map a single input datum (or multiple inputs) into probability distributions and subsequently compute the kernel. In other words, we map points as $x \to p(x|x)$ and $x' \to p(x|x')$. For the setting of $\rho = 1/2$ the kernel is none other than the classical Bhattacharyya similarity measure (Kondor and Jebara, 2003), the affinity corresponding to Hellinger divergence (Jebara and Kondor, 2003).[1]

One goal of this article is to explore a contact point between discriminative learning (support vector machines and kernels) and generative learning (distributions and graphical models). Discriminative learning directly optimizes performance for a given classification or regression task. Meanwhile, generative learning provides a rich palette of tools for exploring models, accommodating unusual input spaces and handling priors. One approach to marrying the two is to estimate parameters in generative models with discriminative learning algorithms and optimize performance on a particular task. Examples of such approaches include conditional learning (Bengio and Frasconi, 1996) or large margin generative modeling (Jaakkola et al., 1999). Another approach is to use kernels to integrate the generative models within a discriminative learning paradigm. The proposed probability product kernel falls into this latter category. Previous efforts to build kernels that accommodate probability distributions include the Fisher kernel (Jaakkola and Haussler, 1998), the heat kernel (Lafferty and Lebanon, 2002) and kernels arising from exponentiating Kullback-Leibler divergences (Moreno et al., 2004). We discuss, compare and contrast these approaches to the probability product kernel in Section 7. One compelling feature of the new kernel is that it is straightforward and efficient to compute over a wide range of distributions and generative models while still producing interesting nonlinear behavior in, for example, support vector machine (SVM) classifiers. The generative models we consider include Gaussians, multinomials, the exponential family, mixture models, hidden Markov models, a wide range of graphical models and (via sampling and mean-field methods) intractable graphical models. The flexibility in choosing a distribution from the wide range of generative models in the field permits the kernel to readily accommodate many interesting input spaces including sequences, counts, graphs, fields and so forth. It also inherits the properties, priors and invariances that may be straightforward to design within a generative modeling paradigm and propagates them into the kernel learning machine. This article thus gives a generative modeling route to kernel design to complement the family of other kernel engineering methods including super-kernels (Ong et al., 2002), convolutional kernels (Haussler, 1999; Collins and Duffy, 2002), alignment kernels (Watkins, 2000), rational kernels (Cortes et al., 2002) and string kernels (Leslie et al., 2002; Vishawanathan and Smola, 2002).

This paper is organized as follows. In Section 2, we introduce the probability product kernel and point out connections to other probabilistic kernels such as Fisher kernels and exponentiated Kullback-Leibler divergences. Estimation methods for mapping points probabilistically to Hilbert

1. This has interesting connections to Kullback-Leibler divergence (Topsoe, 1999).

space are outlined. In Section 3 we elaborate the kernel for the exponential family and show its specific form for classical distributions such as the Gaussian and multinomial. Interestingly, the kernel can be computed for mixture models and graphical models in general and this is elaborated in Section 4. Intractable graphical models are then handled via structured mean-field approximations in Section 6. Alternatively, we describe sampling methods for approximately computing the kernel. Section 7 discusses the differences and similarities between our probability product kernels and previously presented probabilistic kernels. We then show experiments with text data sets and multinomial kernels, with biological sequence data sets using hidden Markov model kernels, and with continuous time series data sets using linear dynamical system kernels. The article then concludes with a brief discussion.

2. A Kernel Between Distributions

Given a positive (semi-)definite kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ on the input space $\mathcal{X}$ and examples $x_1, x_2, \ldots, x_m \in \mathcal{X}$ with corresponding labels $y_1, y_2, \ldots, y_m$, kernel based learning algorithms return a hypothesis of the form $h(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x) + b$. The role of the kernel is to capture our prior assumptions of similarity in $\mathcal{X}$. When $k(x, x')$ is large, we expect that the corresponding labels be similar or the same.

Classically, the inputs $\{x_i\}_{i=1}^{m}$ are often represented as vectors in $\mathbb{R}^n$, and k is chosen to be one of a small number of popular positive definite functions on $\mathbb{R}^n$, such as the Gaussian RBF

$$k(x, x') = e^{-\|x - x'\|^2/(2\sigma^2)}. \tag{1}$$

Once we have found an appropriate kernel, the problem becomes accessible to a whole array of non-parametric discriminative kernel based learning algorithms, such as support vector machines, Gaussian processes, etc. (Schölkopf and Smola, 2002).

In contrast, generative models fit probability distributions $p(x)$ to $x_1, x_2, \ldots, x_m$ and base their predictions on the likelihood under different models. When faced with a discriminative task, approaching it from the generative angle is generally suboptimal. On the other hand, it is often easier to capture the structure of complex objects with generative models than directly with kernels. Specific examples include multinomial models for text documents, hidden Markov models for speech, Markov random fields for images and linear dynamical systems for motion. In the current paper we aim to combine the advantages of both worlds by

1. fitting separate probabilistic models $p_1(x), p_2(x), \ldots p_m(x)$ to $x_1, x_2, \ldots, x_m$;
2. defining a novel kernel $k^{\text{prob}}(p, p')$ between probability distributions on $\mathcal{X}$;
3. finally, defining the kernel between examples to equal k^{prob} between the corresponding distributions:
$$k(x, x') = k^{\text{prob}}(p, p').$$

We then plug this kernel into any established kernel based learning algorithm and proceed as usual. We define k^{prob} in a rather general form and then investigate special cases.

Definition 1 *Let p and p' be probability distributions on a space $\mathcal{X}$ and ρ be a positive constant. Assume that $p^\rho, p'^\rho \in L_2(\mathcal{X})$, i.e. that $\int_{\mathcal{X}} p(x)^{2\rho} dx$ and $\int_{\mathcal{X}} p'(x)^{2\rho} dx$ are well defined (not infinity).*

The **probability product kernel** *between distributions p and p' is defined*

$$k^{prob}(p,p') = \int_{\mathcal{X}} p(x)^{\rho} p'(x)^{\rho}\, dx = \left\langle p^{\rho}, p'^{\rho} \right\rangle_{L_2}. \tag{2}$$

It is well known that $L_2(\mathcal{X})$ is a Hilbert space, hence for any set $\mathcal{P}$ of probability distributions over $\mathcal{X}$ such that $\int_{\mathcal{X}} p(x)^{2\rho} dx$ is finite for any $p \in \mathcal{P}$, the kernel defined by (2) is positive definite. The probability product kernel is also a simple and intuitively compelling notion of similarity between distributions.

2.1 Special Cases and Relationship to Statistical Affinities

For $\rho = 1/2$

$$k(p,p') = \int \sqrt{p(x)}\sqrt{p'(x)}\, dx,$$

which we shall call the **Bhattacharyya kernel**, because in the statistics literature it is known as Bhattacharyya's affinity between distributions (Bhattacharyya, 1943), related to the better-known Hellinger's distance

$$H(p,p') = \frac{1}{2}\int \left(\sqrt{p(x)} - \sqrt{p'(x)}\right)^2 dx$$

by $H(p,p') = \sqrt{2 - 2k(p,p')}$. Hellinger's distance can be seen as a symmetric approximation to the Kullback-Leibler (KL) divergence, and in fact is a bound on KL, as shown in (Topsoe, 1999), where relationships between several common divergences are discussed. The Bhattacharyya kernel was first introduced in (Kondor and Jebara, 2003) and has the important special property $k(x,x) = 1$.

When $\rho = 1$, the kernel takes the form of the expectation of one distribution under the other:

$$k(x,x') = \int p(x)\, p'(x)\, dx = \mathrm{E}_p[p'(x)] = \mathrm{E}_{p'}[p(x)]. \tag{3}$$

We call this the **expected likelihood kernel**.

It is worth noting that when dealing with distributions over discrete spaces $\mathcal{X} = \{x_1, x_2, \ldots\}$, probability product kernels have a simple geometrical interpretation. In this case p can be represented as a vector $p = (p_1, p_2, \ldots)$ where $p_i = \Pr(X = x_i)$. The probability product kernel is then the dot product between $\tilde{p} = \left(p_1^{\rho}, p_2^{\rho}, \ldots\right)$ and $\tilde{p}' = \left(p'^{\rho}_1, p'^{\rho}_2, \ldots\right)$.

In particular, in the case of the Bhattacharyya kernel, $\tilde{p}$ and $\tilde{p}'$ are vectors on the unit sphere. This type of similarity measure is not unknown in the literature. In text recognition, for example, the so-called "cosine-similarity" (i.e. the dot product measure) is well entrenched, and it has been observed that preprocessing word counts by taking square roots often improves performance (Goldzmidt and Sahami, 1998; Cutting et al., 1992). Lafferty and Lebanon also arrive at a similar kernel, but from a very different angle, looking at the diffusion kernel on the statistical manifold of multinomial distributions (Lafferty and Lebanon, 2002).

2.2 Frequentist and Bayesian Methods of Estimation

Various statistical estimation methods can be used to fit the distributions $p_1, p_2, \ldots, p_m$ to the examples $x_1, x_2, \ldots, x_m$. Perhaps it is most immediate to consider a parametric family $\{p_\theta(x)\}$, take the maximum likelihood estimators $\hat{\theta}_i = \arg\max_\theta p_\theta(x_i)$, and set $p_i(x) = p_{\hat{\theta}_i}(x)$.

	$\mathcal{T}(x)$	$\mathcal{A}(x)$	$\mathcal{K}(\theta)$
Gaussian ($\theta=\mu,\ \sigma^2=1$)	x	$-\frac{1}{2}x^Tx-\frac{D}{2}\log(2\pi)$	$\frac{1}{2}\theta^T\theta$
Gaussian ($\theta=\sigma^2,\ \mu=0$)	x	$-\frac{1}{2}\log(2\pi)$	$-\frac{1}{2}\log\theta$
Exponential ($\theta=-1/\beta$)	x	0	$-\log(-\theta)$
Gamma ($\theta=\alpha$)	$\log x$	$-\log x-x$	$\log\Gamma(\theta)$
Poisson ($\theta=\log p$)	x	$-\log(x!)$	$\exp\theta$
Multinomial ($\theta_i=\log\alpha_i$)	$(x_1,x_2,\ldots)$	$\log\big(\big(\sum_i x_i\big)!/\big(\prod_i x_i!\big)\big)$	1

Table 1: Some well-known exponential family distributions

The alternative, Bayesian, strategy is to postulate a prior on θ, invoke Bayes' rule

$$p(\theta|\chi)=\frac{p(\chi|\theta)\,p(\theta)}{\int p(\chi|\theta)\,p(\theta)\,d\theta},$$

and then use either the distribution $p(x|\hat{\theta}_{\mathrm{MAP}})$ based on the maximum a posteriori estimator $\hat{\theta}_{\mathrm{MAP}}=\arg\max_\theta p(\theta|\chi)$, or the true posterior

$$p(x|\chi)=\int p(x|\theta)\,p(\theta|\chi)\,d\theta. \tag{4}$$

At this point the reader might be wondering to what extent it is justifiable to fit distributions to single data points. It is important to bear in mind that $p_i(x)$ is but an intermediate step in forming the kernel, and is not necessarily meant to model anything in and of itself. The motivation is to exploit the way that probability distributions capture similarity: assuming that our model is appropriate, if a distribution fit to one data point gives high likelihood to another data point, this indicates that the two data points are in some sense similar. This is particularly true of intricate graphical models commonly used to model structured data, such as times series, images, etc.. Such models can capture relatively complicated relationships in real world data. Probability product kernels allow kernel methods to harness the power of such models.

When χ is just a point in $\mathbb{R}^n$ with no further structure, probability product kernels are less compelling. Nevertheless, it is worth noting that even the Gaussian RBF kernel (1) can be regarded as a probability product kernel (by setting setting $\rho=1$ and choosing $p_i(x)$ to be a $\sigma/2$ variance Gaussian fit to x_i by maximum likelihood). In particular situations when the point χ does not yield a reliable estimate of p (typically because the class of generative models is too flexible relative to the datum), we may consider regularizing the maximum likelihood estimate of each probability by fitting it to the neighbourhood of a point or by employing other more global estimators of the densities. Other related methods that use points to estimate local distributions including kernel density estimation (KDE) where the global density is composed of local density models centered on each datum (Silverman, 1986). We next discuss a variety of distributions (in particular, the exponential family) where maximum likelihood estimation is well behaved and the probability product kernel is straightforward to compute.

3. Exponential Families

A family of distributions parameterized by $\theta \in \mathbb{R}^D$ is said to form an exponential family if its members are of the form

$$p_\theta(x) \quad = \quad \exp(\mathcal{A}(x) + \theta^T \mathcal{T}(x) - \mathcal{K}(\theta)).$$

Here $\mathcal{A}$ is called the measure, $\mathcal{K}$ is the the cumulant generating function and $\mathcal{T}$ are the sufficient statistics.

Some of the most common statistical distributions, including the Gaussian, multinomial, Poisson and Gamma distributions, are exponential families (Table 1). Note that to ensure that $p_\theta(x)$ is normalized, $\mathcal{A}$, $\mathcal{K}$ and θ must be related through the Laplace transform

$$\mathcal{K}(\theta) = \log \int \exp\left(\mathcal{A}(x) + \theta^T \mathcal{T}(x)\right) dx.$$

The Bhattacharyya kernel ($\rho = 1/2$) can be computed in closed form for any exponential family:

$$\begin{aligned} k(\chi, \chi') = k(p, p') &= \int_x p_\theta(x)^{1/2} p_{\theta'}(x)^{1/2} dx \\ &= \int_x \exp\left(\mathcal{A}(x) + \left(\tfrac{1}{2}\theta + \tfrac{1}{2}\theta'\right)^T \mathcal{T}(x) - \tfrac{1}{2}\mathcal{K}(\theta) - \tfrac{1}{2}\mathcal{K}(\theta')\right) dx \\ &= \exp\left(\mathcal{K}\left(\tfrac{1}{2}\theta + \tfrac{1}{2}\theta'\right) - \tfrac{1}{2}\mathcal{K}(\theta) - \tfrac{1}{2}\mathcal{K}(\theta')\right). \end{aligned}$$

For $\rho \neq 1/2$, $k(\chi, \chi')$ can only be written in closed form when $\mathcal{A}$ and $\mathcal{T}$ obey some special properties. Specifically, if $2\rho\mathcal{A}(x) = \mathcal{A}(\eta x)$ for some η and $\mathcal{T}$ is linear in x, then

$$\log \int \exp\left(2\rho\, \mathcal{A}(x) + \rho\left(\theta + \theta'\right)^T \mathcal{T}(x)\right) dx =$$
$$\log\left[\frac{1}{\eta}\int \exp\left(\mathcal{A}(\eta x) + \frac{\rho}{\eta}\left(\theta + \theta'\right)^T \mathcal{T}(\eta x)\right) d(\eta x)\right] = \mathcal{K}\left(\tfrac{\rho}{\eta}\left(\theta + \theta'\right)\right) - \log \eta,$$

hence

$$k(p, p') = \exp\left[\mathcal{K}\left(\tfrac{\rho}{\eta}\left(\theta + \theta'\right)\right) - \log\eta - \tfrac{\rho}{2}\mathcal{K}(\theta) - \tfrac{\rho}{2}\mathcal{K}(\theta')\right].$$

In the following we look at some specific examples.

3.1 The Gaussian Distribution

The D dimensional Gaussian distribution $p(x) = \mathcal{N}(\mu, \Sigma)$ is of the form

$$p(x) = (2\pi)^{-D/2} |\Sigma|^{-1/2} \exp\left(-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

where Σ is a positive definite matrix and $|\Sigma|$ denotes its determinant. For a pair of Gaussians $p = \mathcal{N}(\mu, \Sigma)$ and $p' = \mathcal{N}(\mu', \Sigma')$, completing the square in the exponent gives the general probability product kernel

$$K_\rho(\chi, \chi') = K_\rho(p, p') = \int_{\mathbb{R}^D} p(x)^\rho p'(x)^\rho dx =$$
$$(2\pi)^{(1-2\rho)D/2} \rho^{-D/2} \left|\Sigma^\dagger\right|^{1/2} \left|\Sigma\right|^{-\rho/2} \left|\Sigma'\right|^{-\rho/2} \exp\left(-\frac{\rho}{2}\left(\mu^T \Sigma^{-1} \mu + \mu'^T \Sigma'^{-1} \mu' - \mu^{\dagger T} \Sigma^\dagger \mu^\dagger\right)\right), \quad (5)$$

where $\Sigma^\dagger = \left(\Sigma^{-1}+\Sigma'^{-1}\right)^{-1}$ and $\mu^\dagger = \Sigma^{-1}\mu+\Sigma'^{-1}\mu'$. When the covariance is isotropic and fixed, $\Sigma = \sigma^2 I$, this simplifies to

$$K_\rho(p,p') = (2\rho)^{-D/2}\,(2\pi\sigma^2)^{(1-2\rho)D/2}\,e^{-\|\mu-\mu'\|^2/(4\sigma^2/\rho)},$$

which, for $\rho = 1$ (the expected likelihood kernel) simply gives

$$k(p,p') = \tfrac{1}{(4\pi\sigma^2)^{D/2}}\,e^{-\|\mu'-\mu\|^2/(4\sigma^2)},$$

recovering, up to a constant factor, the Gaussian RBF kernel (1).

3.2 The Bernoulli Distribution

The Bernoulli distribution $p(x) = \gamma^x(1-\gamma)^{1-x}$ with parameter $\gamma\in(0,1)$, and its D dimensional variant, sometimes referred to as Naive Bayes,

$$p(x) = \prod_{d=1}^{D}\gamma_d^{x_d}(1-\gamma_d)^{1-x_d}$$

with $\gamma\in(0,1)^D$, are used to model binary $x\in\{0,1\}$ or multidimensional binary $x\in\{0,1\}^D$ observations. The probability product kernel

$$K_\rho(\mathcal{x},\mathcal{x}') = K_\rho(p,p') = \sum_{x\in\{0,1\}^D}\prod_{d=1}^{D}(\gamma_d\gamma'_d)^{\rho x_d}((1-\gamma_d)(1-\gamma'_d))^{\rho(1-x_d)}$$

factorizes as

$$K_\rho(p,p') = \prod_{d=1}^{D}\left[(\gamma_d\gamma'_d)^\rho + (1-\gamma_d)^\rho(1-\gamma'_d)^\rho\right].$$

3.3 The Multinomial Distribution

The multinomial model

$$p(x) = \frac{s!}{x_1!x_2!\ldots x_D!}\,\alpha_1^{x_1}\alpha_2^{x_2}\ldots\alpha_D^{x_D}$$

with parameter vector $\alpha = (\alpha_1,\alpha_2,\ldots,\alpha_D)$ subject to $\sum_{d=1}^D\alpha_d = 1$ is commonly used to model discrete integer counts $x = (x_1,x_2,\ldots,x_D)$ with $\sum_{i=1}^D x_i = s$ fixed, such as the number of occurrences of words in a document. The maximum likelihood estimate given observations $x^{(1)},x^{(2)},\ldots,x^{(n)}$ is

$$\hat{\alpha}_d = \frac{\sum_{i=1}^n x_d^{(i)}}{\sum_{i=1}^n\sum_{d=1}^D x_d^{(i)}}.$$

For fixed s, the Bhattacharyya kernel ($\rho = 1/2$) can be computed explicitly using the multinomial theorem:

$$k(p,p') = \sum_{\substack{x=(x_1,x_2,\ldots,x_D)\\ \sum_i x_i = s}}\frac{s!}{x_1!x_2!\ldots x_D!}\prod_{d=1}^{D}(\alpha_d\alpha'_d)^{x_d/2} = \left[\sum_{d=1}^{D}(\alpha_d\alpha'_d)^{1/2}\right]^s \tag{6}$$

which is equivalent to the homogeneous polynomial kernel of order s between the vectors $(\sqrt{\alpha_1}, \sqrt{\alpha_2}, \ldots, \sqrt{\alpha_D})$ and $(\sqrt{\alpha'_1}, \sqrt{\alpha'_2}, \ldots, \sqrt{\alpha'_D})$. When s is not constant, we can sum over all its possible values

$$k(p,p') = \sum_{s=0}^{\infty} \left[\sum_{d=1}^{D} (\alpha_d \alpha'_d)^{1/2} \right]^s = \left(1 - \sum_{d=1}^{D} (\alpha_d \alpha'_d)^{1/2} \right)^{-1}$$

or weight each power differently, leading to a power series expansion.

3.4 The Gamma and Exponential Distributions

The gamma distribution $\Gamma(\alpha,\beta)$ parameterized by $\alpha > 0$ and $\beta > 0$ is of the form

$$p(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta}.$$

The probability product kernel between $p \sim \Gamma(\alpha,\beta)$ and $p' \sim \Gamma(\alpha',\beta')$ will then be

$$k_\rho(p,p') = \frac{\Gamma(\alpha^\dagger)\beta^{\dagger\alpha^\dagger}}{\left[\Gamma(\alpha)\beta^\alpha \Gamma(\alpha')\beta'^{\alpha'}\right]^\rho}$$

where $\alpha^\dagger = \rho(\alpha+\alpha'-2)+1$ and $1/\beta^\dagger = \rho(1/\beta+1/\beta')$. When $\rho = 1/2$ this simplifies to $\alpha^\dagger = (\alpha+\alpha')/2$ and $1/\beta^\dagger = (1/\beta+1/\beta')/2$.

The exponential distribution $p(x) = \frac{1}{\beta} e^{-x/\beta}$ can be regarded as a special case of the gamma family with $\alpha = 1$. In this case

$$k_\rho(p,p') = \frac{\rho(1/\beta+1/\beta')}{(\beta\beta')^\rho}.$$

4. Latent and Graphical Models

We next upgrade beyond the exponential family of distributions and derive the probability product kernel for more versatile generative distributions. This is done by considering latent variables and structured graphical models. While introducing additional complexity in the generative model and hence providing a more elaborate probability product kernel, these models do have the caveat that estimators such as maximum likelihood are not as well-behaved as in the simple exponential family models and may involve expectation-maximization or other only locally optimal estimates. We first discuss the simplest latent variable models, mixture models, which correspond to a single unobserved discrete parent of the emission and then upgrade to more general graphical models such as hidden Markov models.

Latent variable models identify a split between the variables in x such that some are observed and are denoted using x_o (where the lower-case 'o' is short for 'observed') while others are hidden and denoted using x_h (where the lower-case 'h' is short for 'hidden'). In the latent case, the probability product kernel is computed only by the inner product between two distributions $p(x_o)$ and $p'(x_o)$ over the observed components, in other words $k(p,p') = \int p(x_o)p'(x_o)dx_o$. The additional variables x_h are incomplete observations that are not part of the original sample space (where the datum or data points to kernelize exist) but an augmentation of it. The desired $p(x_o)$ therefore, involves marginalizing away the hidden variables:

$$p(x_o) \;=\; \sum_{x_h} p(x_o, x_h) \;=\; \sum_{x_h} p(x_h)p(x_o|x_h).$$

For instance, we may consider a mixture of exponential families model (a direct generalization of the exponential family model) where x_h is a single discrete variable and where $p(x_\mathrm{o}|x_\mathrm{h})$ are exponential family distributions themselves. The probability product kernel is then given as usual between two such distributions $p(x_\mathrm{o})$ and $p'(x_\mathrm{o})$ which expand as follows:

$$k(p,p') = \sum_{x_\mathrm{o}} (p(x))^\rho \left(p'(x)\right)^\rho = \sum_{x_\mathrm{o}} \left(\sum_{x_\mathrm{h}} p(x_\mathrm{h}) p(x_\mathrm{o}|x_\mathrm{h}) \right)^\rho \left(\sum_{x'_\mathrm{h}} p'(x'_\mathrm{h}) p'(x_\mathrm{o}|x'_\mathrm{h}) \right)^\rho .$$

We next note a slight modification to the kernel which makes latent variable models tractable when $\rho \neq 1$. This alternative kernel is denoted $\tilde{k}(p,p')$ and also satisfies Mercer's condition using the same line of reasoning as was employed for $k(p,p')$ in the previous sections. Essentially, for $\tilde{k}(p,p')$ we will assume that the power operation involving ρ is performed on each entry of the joint probability distribution $p(x_\mathrm{o},x_\mathrm{h})$ instead of on the marginalized $p(x_\mathrm{o})$ alone as follows:

$$\tilde{k}(p,p') = \sum_{x_\mathrm{o}} \sum_{x_\mathrm{h}} (p(x_\mathrm{h}) p(x_\mathrm{o}|x_\mathrm{h}))^\rho \sum_{x'_\mathrm{h}} \left(p'(x'_\mathrm{h}) p'(x_\mathrm{o}|x'_\mathrm{h})\right)^\rho .$$

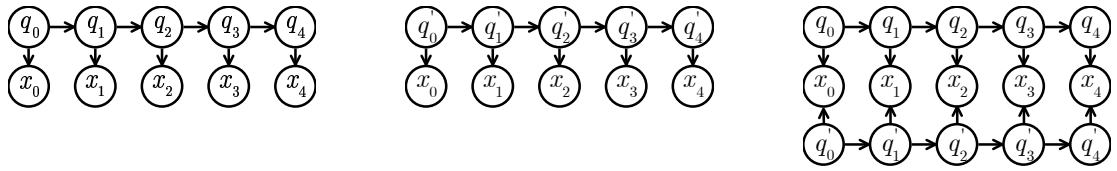
While the original kernel $k(p,p')$ involved the mapping to Hilbert space given by $\mathcal{X} \rightarrow p(x_\mathrm{o})$, the above $\tilde{k}(p,p')$ corresponds to mapping each datum to an augmented distribution over a more complex marginalized space where probability values are squashed (or raised) by a power of ρ. Clearly, $k(p,p')$ and $\tilde{k}(p,p')$ are formally equivalent when $\rho = 1$. However, for other settings of ρ, we prefer handling latent variables with $\tilde{k}(p,p')$ (and at times omit the $\sim$ symbol when it is self-evident) since it readily accommodates efficient marginalization and propagation algorithms (such as the junction tree algorithm (Jordan and Bishop, 2004)).

One open issue with latent variables is that the $\tilde{k}$ kernels that marginalize over them may produce different values if the underlying latent distributions have different joint distributions over hidden and observed variables even though the marginal distribution over observed variables stays the same. For instance, we may have a two-component mixture model for p with both components having the same identical emission distributions yet the kernel $\tilde{k}(p,p')$ evaluates to a different value if we collapse the identical components in p into one emission distribution. This is to be expected since the different latent components imply a slightly different mapping to Hilbert space.

We next describe the kernel for various graphical models which essentially impose a factorization on the joint probability distribution over the N variables x_o and x_h. This article focuses on directed graphs (although undirected graphs can also be kernelized) where this factorization implies that the joint distribution can be written as a product of conditionals over each node or variable $x_i \in \{x_\mathrm{o}, x_\mathrm{h}\}, i = 1 \ldots N$ given its parent nodes or set of parent variables $x_{pa(i)}$ as $p(x_\mathrm{o},x_\mathrm{h}) = \prod_{i=1}^N p(x_i|x_{pa(i)})$. We now discuss particular cases of graphical models including mixture models, hidden Markov models and linear dynamical systems.

4.1 Mixture Models

In the simplest scenario, the latent graphical model could be a mixture model, such as a mixture of Gaussians or mixture of exponential family distributions. Here, the hidden variable x_h is a single discrete variable which is a parent of a single x_o emission variable in the exponential family. To derive the full kernel for the mixture model, we first assume it is straightforward to compute an



(a) HMM for p. (b) HMM for p'. (c) Combined graphs.

Figure 1: Two hidden Markov models and the resulting graphical model as the kernel couples common parents for each node creating undirected edges between them.

elementary kernel between any pair of entries (indexed via $x_{\rm h}$ and $x'_{\rm h}$) from each mixture model as follows:

$$\tilde{k}(p(x_{\rm o}|x_{\rm h}), p'(x_{\rm o}|x'_{\rm h})) \;=\; \sum_{x_{\rm o}} \left(p(x_{\rm o}|x_{\rm h})p'(x_{\rm o}|x'_{\rm h})\right)^{\rho}.$$

In the above, the conditionals could be, for instance, exponential family (emission) distributions. Then, we can easily compute the kernel for a mixture model of such distributions using these elementary kernel evaluations and a weighted summation of all possible pairings of components of the mixtures:

$$\begin{aligned}\tilde{k}(p,p') &= \sum_{x_{\rm o}}\sum_{x_{\rm h}} (p(x_{\rm h})p(x_{\rm o}|x_{\rm h}))^{\rho} \sum_{x'_{\rm h}} \left(p'(x'_{\rm h})p'(x_{\rm o}|x'_{\rm h})\right)^{\rho} \\ &= \sum_{x_{\rm h},x'_{\rm h}} \left(p(x_{\rm h})p'(x'_{\rm h})\right)^{\rho} \tilde{k}(p(x_{\rm o}|x_{\rm h}), p'(x_{\rm o}|x'_{\rm h})).\end{aligned}$$

Effectively, the mixture model kernel involves enumerating all settings of the hidden variables $x_{\rm h}$ and $x'_{\rm h}$. Taking the notation $|.|$ to be the cardinality of a variable, we effectively need to perform and sum a total of $|x_{\rm h}| \times |x'_{\rm h}|$ elementary kernel computations $\tilde{k}_{x_{\rm h},x'_{\rm h}}$ between the individual emission distributions.

4.2 Hidden Markov Models

We next upgrade beyond mixtures to graphical models such as hidden Markov models (HMMs). Considering hidden Markov models as the generative model p allows us to build a kernel over sequences of variable lengths. However, HMMs and many popular Bayesian networks have a large number of (hidden and observed) variables and enumerating all possible hidden states for two distributions p and p' quickly becomes inefficient since $x_{\rm h}$ and $x'_{\rm h}$ are multivariate and/or have large cardinalities. However, unlike the plain mixture modeling case, HMMs have an efficient graphical model structure implying a factorization of their distribution which we can leverage to compute our kernel efficiently. A hidden Markov model over sequences of length $T+1$ has observed variables $x_{\rm o} = \{x_0, \ldots, x_T\}$ and hidden states $x_{\rm h} = \{q_0, \ldots, q_T\}$. The graphical model for an HMM in Figure 1(a) reflects its Markov chain assumption and leads to the following probability density function (note here we define $q_{-1} = \{\}$ as a null variable for brevity or, equivalently, $p(q_0|q_{-1}) = p(q_0)$):

$$p(x_{\rm o}) \;=\; \sum_{x_{\rm h}} p(x_{\rm o}, x_{\rm h}) \;=\; \sum_{q_0} \cdots \sum_{q_T} \prod_{t=0}^{T} p(x_t|q_t)p(q_t|q_{t-1}).$$

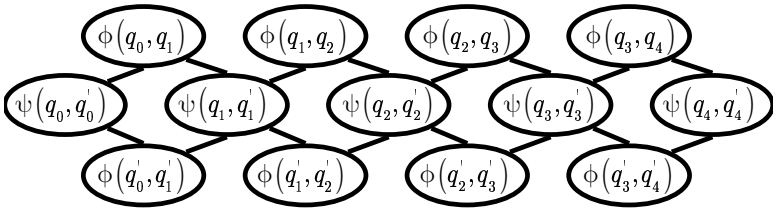


Figure 2: The undirected clique graph obtained from the kernel for efficiently summing over both sets of hidden variables x_{h} and x'_{h}.

To compute the kernel $\tilde{k}(p, p')$ in a brute force manner, we need to sum over all configurations of $q_0, \ldots, q_T$ and $q'_0, \ldots, q'_T$ while computing for each each configuration its corresponding elementary kernel $\tilde{k}(p(x_0, \ldots, x_T|q_0, \ldots, q_T), p'(x_0, \ldots, x_T|q'_0, \ldots, q'_T))$. Exploring each setting of the state space of the HMMs (shown in Figure 1(a) and (b)) is highly inefficient requiring $|q_t|^{(T+1)} \times |q'_t|^{(T+1)}$ elementary kernel evaluations. However, we can take advantage of the factorization of the hidden Markov model by using graphical modeling algorithms such as the junction tree algorithm (Jordan and Bishop, 2004) to compute the kernel efficiently. First, we use the factorization of the HMM in the kernel as follows:

$$\begin{aligned}
\tilde{k}(p, p') &= \sum_{x_0,\ldots,x_T} \sum_{q_0 \ldots q_T} \prod_{t=0}^{T} p(x_t|q_t)^\rho p(q_t|q_{t-1})^\rho \sum_{q'_0 \ldots q'_T} \prod_{t=0}^{T} p'(x_t|q'_t)^\rho p(q'_t|q'_{t-1})^\rho \\
&= \sum_{q_0 \ldots q_T} \sum_{q'_0 \ldots q'_T} \prod_{t=0}^{T} p(q_t|q_{t-1})^\rho p(q'_t|q'_{t-1})^\rho \left(\sum_{x_t} p(x_t|q_t)^\rho p'(x_t|q'_t)^\rho \right) \\
&= \sum_{q_0 \ldots q_T} \sum_{q'_0 \ldots q'_T} \prod_{t=0}^{T} p(q_t|q_{t-1})^\rho p(q'_t|q'_{t-1})^\rho \psi(q_t, q'_t) \\
&= \sum_{q_T} \sum_{q'_T} \psi(q_T, q'_T) \prod_{t=1}^{T} \sum_{q_{t-1}} \sum_{q'_{t-1}} p(q_t|q_{t-1})^\rho p(q'_t|q'_{t-1})^\rho \psi(q_{t-1}, q'_{t-1}) p(q_0)^\rho p'(q'_0)^\rho.
\end{aligned}$$

In the above we see that we only need to compute the kernel for each setting of the hidden variable for a given time step on its own. Effectively, the kernel couples the two HMMs via their common children as in Figure 1(c). The kernel evaluations, once solved, form non-negative clique potential functions $\psi(q_t, q'_t)$ with a different setting for each value of q_t and q'_t. These couple the hidden variable of each HMM for each time step. The elementary kernel evaluations are easily computed (particularly if the emission distributions $p(x_t|q_t)$ are in the exponential family) as:

$$\tilde{k}(p(x_t|q_t), p'(x_t|q'_t)) = \psi(q_t, q'_t) = \sum_{x_t} p(x_t|q_t)^\rho p'(x_t|q'_t)^\rho.$$

Only a total of $(T+1) \times |q_t| \times |q'_t|$ such elementary kernels need to be evaluated and form a limited number of such clique potential functions. Similarly, each conditional distribution $p(q_t|q_{t-1})^\rho$ and $p'(q'_t|q'_{t-1})^\rho$ can also be viewed as a non-negative clique potential function, i.e. $\phi(q_t, q_{t-1}) =$

$p(q_t|q_{t-1})^\rho$ and $\phi(q'_t, q'_{t-1}) = p'(q'_t|q'_{t-1})^\rho$. Computing the kernel then involves marginalizing over the hidden variables x_h, which is done by running a forward-backward or junction-tree algorithm on the resulting undirected clique tree graph shown in Figure 2. Thus, to compute the kernel for two sequences $\mathcal{x}$ and $\mathcal{x}'$ of lengths $T_\mathcal{x}+1$ and $T_{\mathcal{x}'}+1$, we train an HMM from each sequence using maximum likelihood and then compute the kernel using the learned HMM transition matrix and emission models for a user-specified sequence length $T+1$ (where T can be chosen according to some heuristic, for instance, the average of all $T_\mathcal{x}$ in the training data set). The following is pseudo-code illustrating how to compute the $\tilde{k}(p,p')$ kernel given two hidden Markov models (p,p') and user-specified parameters T and ρ.

$$
\begin{aligned}
&\Phi(q_0, q'_0) = p(q_0)^\rho p'(q'_0)^\rho \\
&\text{for } t = 1 \ldots T \\
&\quad \Phi(q_t, q'_t) = \sum_{q_{t-1}} \sum_{q'_{t-1}} p(q_t|q_{t-1})^\rho p'(q'_t|q'_{t-1})^\rho \psi(q_{t-1}, q'_{t-1}) \Phi(q_{t-1}, q'_{t-1}) \\
&\text{end} \\
&\tilde{k}(p,p') = \sum_{q_T} \sum_{q'_T} \Phi(q_T, q'_T) \psi(q_T, q'_T).
\end{aligned}
$$

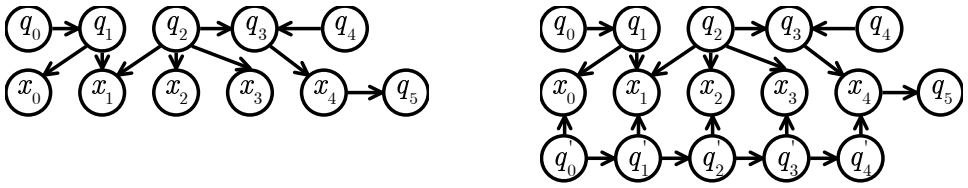
Note that the hidden Markov models p and p' need not have the same number of hidden states for the kernel computation.

4.3 Bayesian Networks

The above method can be applied to Bayesian networks in general where hidden and observed variables factorize according to $p(x) = \prod_{i=1}^N p(x_i|x_{pa(i)})$. The general recipe mirrors the above derivation for the hidden Markov model case. In fact, the two Bayesian networks need not have the same structure as long as they share the same sample space over the emission variables x_o. For instance, consider computing the kernel between the Bayesian network in Figure 3(a) and the hidden Markov model in Figure 1(b) over the same sample space. The graphs can be connected with their common children as in Figure 3(b). The parents with common children are married and form cliques of hidden variables. We then only need to evaluate the elementary kernels over these cliques giving the following non-negative potential functions for each observed node (for instance $x_i \in x_\mathrm{o}$) under each setting of all its parents' nodes (from both p and p'):

$$
\begin{aligned}
\psi(x_{\mathrm{h},\mathrm{pa}(i)}, x'_{\mathrm{h}',\mathrm{pa}(i)}) &= \tilde{k}(p(x_i|x_{\mathrm{h},\mathrm{pa}(i)}), p'(x_i|x_{\mathrm{h}',\mathrm{pa}(i)'})) \\
&= \sum_{x_i} p(x_i|x_{\mathrm{h},\mathrm{pa}(i)})^\rho p'(x_i|x_{\mathrm{h}',\mathrm{pa}(i)'})^\rho.
\end{aligned}
$$

After marrying common parents, the resulting clique graph is then built and used with a junction tree algorithm to compute the overall kernel from the elementary kernel evaluations. Although the resulting undirected clique graph may not always yield a dramatic improvement in efficiency as was seen in the hidden Markov model case, we note that propagation algorithms (loopy or junction tree algorithms) are still likely to provide a more efficient evaluation of the kernel than the brute force enumeration of all latent configurations of both Bayesian networks in the kernel (as was described in the generic mixture model case). While the above computations emphasized discrete latent variables, the kernel is also computable over continuous latent configuration networks, where



(a) Bayesian network for p. (b) Combined graphs.

Figure 3: The resulting graphical model from a Bayesian network and a hidden Markov model as the kernel couples common parents for each node creating undirected edges between them and a final clique graph for the junction tree algorithm.

x_{h} contains scalars and vectors, as is the case in linear Gaussian models, which we develop in the next subsections.

4.4 Linear Gaussian Models

A linear Gaussian model on a directed acyclic graph $\mathcal{G}$ with variables $x_1, x_2, \ldots, x_N$ associated to its vertices is of the form

$$p(x_1, x_2, \ldots, x_N) = \prod_i \mathcal{N}\left(x_i ; \beta_{i,0} + \textstyle\sum_{j \in \mathrm{pa}(i)} \beta_{i,j} x_j, \Sigma_i\right)$$

where $\mathcal{N}(x; \mu, \Sigma)$ is the multivariate Gaussian distribution, and $\mathrm{pa}(i)$ is the index set of parent vertices associated with the i^{th} vertex. The unconditional joint distribution can be recovered from the conditional form and is itself a Gaussian distribution over the variables in the model $p(x_1, x_2, \ldots, x_N)$. Hence, the probability product kernel between a pair of linear Gaussian models can be computed in closed form by using the unconditional joint distribution and (5).

Latent variable models require an additional marginalization step. Variables are split into two sets: the observed variables x_{o} and the hidden variables x_{h}. After the joint unconditional distribution $p(x_{\mathrm{o}}, x_{\mathrm{h}})$ has been computed, the hidden variables can be trivially integrated out:

$$k(p, p') = \int \left(\int p(x_{\mathrm{o}}, x_{\mathrm{h}})\, dx_{\mathrm{h}} \right)^{\rho} \left(\int p(x_{\mathrm{o}}, x'_{\mathrm{h}})\, dx'_{\mathrm{h}} \right)^{\rho} dx_{\mathrm{o}} = \int p(x_{\mathrm{o}})^{\rho}\, p'(x_{\mathrm{o}})^{\rho}\, dx_{\mathrm{o}},$$

so that the kernel is computed between Gaussians over only the observed variables.

4.5 Linear Dynamical Systems

A commonly used linear Gaussian model with latent variables is the linear dynamical system (LDS) (Shumway and Stoffer, 1982), also known as the state space model or Kalman filter model. The LDS is the continuous state analog of the hidden Markov model and shares the same conditional independence graph Figure 1(a). Thus, it is appropriate for modeling continuous time series data and continuous dynamical systems. The LDS's joint probability model is of the form

$$p(x_0, \ldots x_T, s_0, \ldots s_T) = \mathcal{N}(s_0; \mu, \Sigma)\, \mathcal{N}(x_0; C s_0, R) \prod_{t=1}^{T} \mathcal{N}(s_t; A s_{t-1}, Q)\, \mathcal{N}(x_t; C s_t, R),$$

where x_t is the observed variable at time t, and s_t is the latent state space variable at time t, obeying the Markov property.

The probability product kernel between LDS models is

$$k(p,p') = \int \mathcal{N}(x;\mu_x,\Sigma_{xx})^\rho \, \mathcal{N}(x;\mu'_x,\Sigma'_{xx})^\rho \, dx,$$

where μ_x and Σ_{xx} are the unconditional mean and covariance, which can be computed from the recursions

$$\begin{aligned} \mu_{s_t} &= A\mu_{s_{t-1}} \\ \mu_{x_t} &= C\mu_{s_t} \\ \Sigma_{s_t s_t} &= A\Sigma_{s_{t-1}s_{t-1}}A' + Q \\ \Sigma_{x_t x_t} &= C\Sigma_{s_t s_t}C' + R. \end{aligned}$$

As with the HMM, we need to set the number of time steps before computing the kernel. Note that the most algorithmically expensive calculation when computing the probability product kernel between Gaussians is taking the inverse of the covariance matrices. The dimensionality of the covariance matrix will grow linearly with the number of time steps and the running time of the matrix inverse grows cubically with the dimensionality. However, the required inverses can be efficiently computed because Σ_{xx} is block diagonal. Each extra time step added to the kernel will simply add another block, and the inverse of a block diagonal matrix can be computed by inverting the blocks individually. Therefore, the running time of inverting the block diagonal covariance matrix will only grow linearly with the number of time steps T and cubically with the (small) block size.

5. Sampling Approximation

To capture the structure of real data, generative models often need to be quite intricate. Closed form solutions for the probability product kernel are then unlikely to exist, forcing us to rely on approximation methods to compute $k(p,p')$.

Provided that evaluating the likelihood $p(x)$ is computationally feasible, and that we can also efficiently sample from the model, in the $\rho = 1$ case (expected likelihood kernel) we may employ the Monte Carlo estimate

$$k(p,p') \approx \frac{\beta}{N}\sum_{i=1}^{N} p'(x^i) + \frac{(1-\beta)}{N'}\sum_{i=1}^{N} p(x'^i),$$

where $x^1,\ldots,x^N$ and $x'^1,\ldots,x'^N$ are i.i.d. samples from p and p' respectively, and $\beta \in [0,1]$ is a parameter of our choosing. By the law of large numbers, this approximation will converge to the true kernel value in the limit $N \to \infty$ for any β. Other sampling techniques such as importance sampling and a variety of flavors of Markov chain Monte Carlo (MCMC), including Gibbs sampling, can be used to improve the rate of the sampling approximation's convergence to the true kernel.

In some cases it may be possible to use the sampling approximation for an general ρ. This is possible if a normalizing factor Z can be computed to renormalize $p(x)^\rho$ into a valid distribution. Z is computable for most discrete distributions and some continuous distribution, such as the Gaussian.

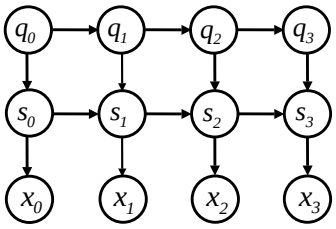


Figure 4: Graph for the Switching Linear Dynamical System.

The sampling approximation then becomes

$$k(p,p') \approx \frac{\beta}{N}\sum_{i=1}^{N} Z\, p'(\hat{x}^i)^\rho + \frac{(1-\beta)}{N'}\sum_{i=1}^{N} Z'\, p(\hat{x}'^i)^\rho,$$

where Z and Z' are the normalizers of p and p' after they are taken to the power of ρ. In this formulation $\hat{x}^1,\ldots,\hat{x}^N$ and $\hat{x}'^1,\ldots,\hat{x}'^N$ are i.i.d. samples from $\hat{p}$ and $\hat{p}'$ where $\hat{p} = \frac{p^\rho}{Z}$ and $\hat{p}' = \frac{p'^\rho}{Z'}$.

In practice, for a finite number of samples, the approximate kernel may not be a valid Mercer kernel. The non-symmetric aspect of the approximation can be alleviated by computing only the lower triangular entries in the kernel matrix and replicating them in the upper half. Problems associated with the approximation not being positive definite can be rectified by either using more samples, or by using an implementation for the kernel based classifier algorithm that can converge to a local optimum such as sequential minimal optimization (Platt, 1999).

5.1 Switching Linear Dynamical Systems

In some cases, part of the model can be evaluated by sampling, while the rest is computed in closed form. This is true for the switching linear dynamical system (SLDS) (Pavlovic et al., 2000), which combines the discrete hidden states of the HMM and the continuous state space of the LDS:

$$p(x,s,q) = p(q_0)\,p(s_0|q_0)\,p(x_0|s_0)\prod_{t=1}^{T} p(q_t|q_{t-1})\,p(s_t|s_{t-1},q_t)\,p(x_t|s_t),$$

where q_t is the discrete hidden state at time t, s_t is the continuous hidden state, and x_t are observed emission variables (Figure 4). Sampling $q^1\ldots,q^N$ according to $p(q)$ and $q'^1,\ldots,q'^N$ according to $p'(q')$, the remaining factors in the corresponding sampling approximation

$$\frac{1}{N}\sum_{i=1}^{N}\int\left(\int p(s_0|q_0^i)\,p'(s_0'|q_0'^i)\prod_{t=1}^{T} p(s_t|s_{t-1},q_t^i)\,p'(s_t'|s_{t-1}',q_t'^i)\prod_{t=0}^{T} p(x_t|s_t)\,p'(x_t|s_t')\,ds\,ds'\right)^\rho dx$$

form a non-stationary linear dynamical system. Once the joint distribution is recovered, it can be evaluated in closed form by (5) as was the case with a simple LDS. For $\rho \neq 1$ this is a slightly different formulation than the sampling approximation in the previous section. This hybrid method has the nice property that it is always a valid kernel for any number of samples since it simply becomes a convex combination of kernels between LDSs. An SLDS model is useful for describing input sequences that have a combination of discrete and continuous dynamics, such as a time series of human motion containing continuous physical dynamics and kinematics variables as well as discrete action variables (Pavlovic et al., 2000).

6. Mean Field Approximation

Another option when the probability product kernel between graphical models cannot be computed in closed form is to use variational bounds, such as the mean field approximation (Jaakkola, 2000). In this method, we introduce a variational distribution $Q(x)$, and using Jensen's inequality, lower bound the kernel:

$$\begin{aligned} k(p,p') &= \int p(x)^\rho \, p'(x)^\rho \, dx = \int \Psi(x)^\rho \, dx = \exp\left(\log \int \frac{Q(x)}{Q(x)} \Psi(x)^\rho \, dx\right) \\ &\geq \exp\left(\int Q(x) \log \frac{\Psi(x)^\rho}{Q(x)} \, dx\right) = \exp\left(\rho \, \mathrm{E}_Q\left[\log \Psi(x)\right] + H(Q)\right) = B(Q), \end{aligned}$$

where $\Psi(x) = p(x)\, p'(x)$ is called the potential, $\mathrm{E}_Q[\,\cdot\,]$ denotes the expectation with respect to $Q(x)$, and $H(Q)$ is the entropy. This transforms the problem from evaluating an intractable integral into that of finding the sufficient statistics of $Q(x)$ and computing the subsequent tractable expectations. Note, using the mean field formulation gives a different (and arguably closer result) to the desired intractable kernel than simply computing the probability product kernels directly between the approximate simple distributions. It is interesting to note the following form of the bound when it is expressed in terms of the Kullback-Leibler divergence, $D(Q\|p) = \int Q(x) \log \frac{Q(x)}{p(x)} dx$, and an entropy term. The bound on the probability product kernel is a function of the Kullback-Leibler divergence to p and p':

$$k(p,p') \;\geq\; B(Q) \;=\; \exp\left(-\rho \, D(Q\|p) - \rho \, D(Q\|p') + (1-2\rho)\, H(Q)\right).$$

The goal is to define $Q(x)$ in a way that makes computing the sufficient statistics easy. When the graphical structure of $Q(x)$ has no edges, which corresponds to the case that the underlying variables are independent, this is called the mean field method. When $Q(x)$ is made up of a number of more elaborate independent structures, it is called the structured mean field method. In either case, the variational distribution factors in the form $Q_1(x_1)Q_2(x_2)\ldots Q_N(x_N)$ with $x_1, x_2, \ldots, x_N$ disjoint subsets of the original variable set x.

Once the structure of $Q(x)$ has been set, its (locally) optimal parameterization is found by cycling through the distributions $Q_1(x_1), Q_2(x_2), \ldots, Q_N(x_N)$ and maximizing the lower bound $B(Q)$ with respect to each one, holding the others fixed:

$$\frac{\partial \log B(Q)}{\partial Q_n(x_n)} = \frac{\partial}{\partial Q_n(x_n)} \left[\rho \int Q_n(x_n) \, \mathrm{E}_Q\left[\log \Psi(x) | x_n\right] dx_n + H(Q_n) + \text{constant}\right] = 0.$$

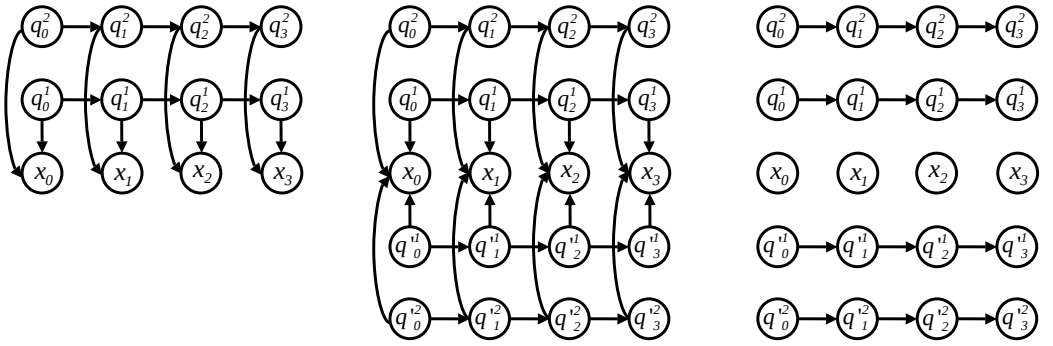
This leads to the update equations

$$Q_n(x_n) = \frac{1}{Z_n} \exp\left(\rho \, \mathrm{E}_Q\left[\log \Psi(x) | x_n\right]\right),$$

where the conditional expectation $\mathrm{E}_Q[\,\cdot\,|\,x_n]$ is computed by taking the expectation over all variables except x_n and

$$Z_n = \int \exp\left(\rho \, \mathrm{E}_Q\left[\log \Psi(x) | x_n\right]\right) dx$$

is the normalizing factor guaranteeing that $Q(x)$ remains a valid distribution.



(a) FHMM $P(x,q)$. (b) Potential $\Psi(x,q,q')$. (c) Variational $Q(x,q,q')$.

Figure 5: Graphs associated with the factorial hidden Markov model, its probability product kernel potential, and its structured mean field distribution.

In practice this approximation may not give positive definite (PD) kernels. Much current research has been focused on kernel algorithms using non-positive definite kernels. Algorithms such as sequential minimal optimization (Platt, 1999) can converge to locally optimal solutions. The learning theory for non-PD kernels is currently of great interest. See Ong et al. (2004) for an interesting discussion and additional references.

6.1 Factorial Hidden Markov Models

One model that traditionally uses a structured mean field approximation for inference is the factorial hidden Markov model (FHMM)(Ghahramani and Jordan, 1997). Such a model is appropriate for modeling output sequences that are emitted by multiple independent hidden processes. The FHMM is given by

$$p(x_0,x_1,x_2,\ldots,x_T) = \prod_{c=1}^{C}\left[p(q_0^c)\prod_{t=1}^{T}p(q_t^c|q_{t-1}^c)\right]\prod_{t=0}^{T}p(x_t|q_t^1,\ldots,q_t^C),$$

where q_t^c is the factored discrete hidden state at time t for chain c and x_t is the observed Gaussian emission variable at time t. The graphs of the joint FHMM distribution, the potential $\Psi(x,q,q')$, and the variational distribution $Q(x,q,q')$ are depicted in Figure 5. The structured mean field approximation is parameterized as

$$\begin{aligned}
Q(q_0^c=i) &\propto \exp\left(\rho\log\phi^c(i) - \frac{\rho}{2}\log|\Sigma_i| - \frac{\rho}{2}\mathrm{E}_Q\left[(x_0-\mu_i)^T\Sigma_i^{-1}(x_0-\mu_i)\right]\right)\\
Q(q_t^c=i\,|\,q_{t-1}^c=j) &\propto \exp\left(\rho\log\Phi^c(i,j) - \frac{\rho}{2}\log|\Sigma_i| - \frac{\rho}{2}\mathrm{E}_Q\left[(x_t-\mu_i)^T\Sigma_i^{-1}(x_t-\mu_i)\right]\right)\\
Q(x_t) &= \mathcal{N}(x_t;\hat{\mu}_t,\hat{\Sigma}_t)\\
\hat{\Sigma}_t &= \frac{1}{\rho}\left[\sum_{i,c}\mathrm{E}_Q\left[s_t^c(i)\right]\Sigma_i^{-1} + \sum_{i,c}\mathrm{E}_Q\left[q_t'^c(i)\right]\Sigma_i'^{-1}\right]^{-1}\\
\hat{\mu}_t &= \rho\hat{\Sigma}_t\left[\sum_{i,c}\mathrm{E}_Q\left[q_t^c(i)\right]\Sigma_i^{-1}\mu_i + \sum_{i,c}\mathrm{E}_Q\left[q_t'^c(i)\right]\Sigma_i'^{-1}\mu_i'\right].
\end{aligned}$$

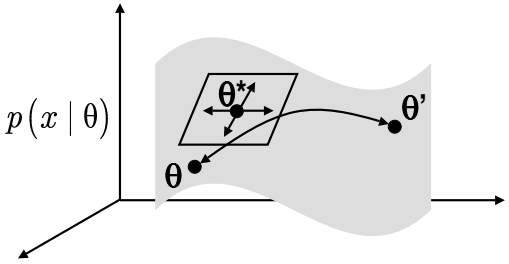


Figure 6: A statistical manifold with a geodesic path between two generative models θ and θ' as well as a local tangent space approximation at, for instance, the maximum likelihood model θ^* for the aggregated data set.

It is necessary to compute the expected sufficient statistics to both update the mean field equations and to evaluate the bound. The expectations $\mathrm{E}_Q[x_t]$ and $\mathrm{E}_Q\left[x_t x_t^T\right]$ can easily be computed from $Q(x_t)$, whereas $Q(s_t^c = i)$ and $Q(s_t^c = i, s_{t-1}^c = j)$ can be computed efficiently by means of a junction tree algorithm on each independent chain in the approximating distribution.

7. Relationship to Other Probabilistic Kernels

While the probability product kernel is not the only kernel to leverage generative modeling, it does have advantages in terms of computational feasibility as well as in nonlinear flexibility. For instance, heat kernels or diffusion kernels on statistical manifolds (Lafferty and Lebanon, 2002) are a more elegant way to generate a kernel in a probabilistic setting, but computations involve finding geodesics on complicated manifolds and finding closed form expressions for the heat kernel, which are only known for simple structures such as spheres and hyperbolic surfaces. Figure 6 depicts such a statistical manifold. The heat kernel can sometimes be approximated via the geodesic distance from p which is parameterized by θ to p' parameterized by θ'. But, we rarely have compact closed-form expressions for the heat kernel or for these geodesics for general distributions. Only Gaussians with spherical covariance and multinomials are readily accommodated. Curiously, the probabilistic product kernel expression for both these two distributions seems to coincide closely with the more elaborate heat kernel form. For instance, in the multinomial case, both involve an inner product between the square-rooted count frequencies. However, the probability product kernel is straightforward to compute for a much wider range of distributions providing a practical alternative to heat kernels.

Another probabilistic approach is the Fisher kernel (Jaakkola and Haussler, 1998) which approximates distances and affinities on the statistical manifold by using the local metric at the maximum likelihood estimate θ^* of the whole data set as shown in Figure 6. One caveat, however, is that this approximation can produce a kernel that is quite different from the exact heat kernel above and may not preserve the interesting nonlinearities implied by the generative model. For instance, in the exponential family case, all distributions generate Fisher kernels that are linear in the sufficient statistics. Consider the Fisher kernel

$$k(x, x') \quad = \quad U_x I_{\theta^*}^{-1} U_{x'}$$

where $I_{\theta^*}^{-1}$ is the inverse Fisher information matrix at the maximum likelihood estimate and where we define

$$U_\chi \quad = \quad \nabla_\theta \log p(\chi|\theta)|_{\theta^*}.$$

In exponential families, $p_\theta(x) = \exp(\mathcal{A}(x) + \theta^T \mathcal{T}(x) - \mathcal{K}(\theta))$ producing the following $U_\chi = \mathcal{T}(\chi) - \mathcal{G}(\theta^*)$ where we define $\mathcal{G}(\theta^*) = \nabla_\theta \mathcal{K}(\theta))|_{\theta^*}$. The resulting Fisher kernel is then

$$k(\chi, \chi') \quad = \quad (\mathcal{T}(\chi) - \mathcal{G}(\theta^*))^T I_{\theta^*}^{-1} \left(\mathcal{T}(\chi')^T - \mathcal{G}(\theta^*)\right),$$

which is an only slightly modified form of the linear kernel in $\mathcal{T}(\chi)$. Therefore, via the Fisher kernel method, a Gaussian distribution over means generates only linear kernels. A Gaussian distribution over means and covariances generates quadratic kernels. Furthermore, multinomials generate log-counts unlike the square root of the counts in the probability product kernel and the heat kernel. In practical text classification applications, it is known that square root squashing functions on frequencies and count typically outperform logarithmic squashing functions (Goldzmidt and Sahami, 1998; Cutting et al., 1992). In (Tsuda et al., 2002; Kashima et al., 2003), another related probabilistic kernel is put forward, the so-called marginalized kernel. This kernel is again similar to the Fisher kernel as well as the probability product kernel. It involves marginalizing a kernel weighted by the posterior over hidden states given the input data for two different points, in other words the output kernel $k(x,x') = \sum_h \sum_{h'} p(h|x)p(h'|x')k((x,h),(x',h'))$. The probability product kernel is similar, however, it involves an inner product over both hidden variables and the input space x with a joint distribution.

A more recently introduced kernel, the exponentiated symmetrized Kullback-Leibler (KL) divergence (Moreno et al., 2004) is also related to the probability product kernel. This kernel involves exponentiating the negated symmetrized KL-divergence between two distributions

$$k(p,p') \quad = \quad \exp(-\alpha D(p\|p') - \alpha D(p'\|p) + \beta) \text{ where } D(p\|p') \; = \; \int_x p(x) \log \frac{p(x)}{p'(x)} dx$$

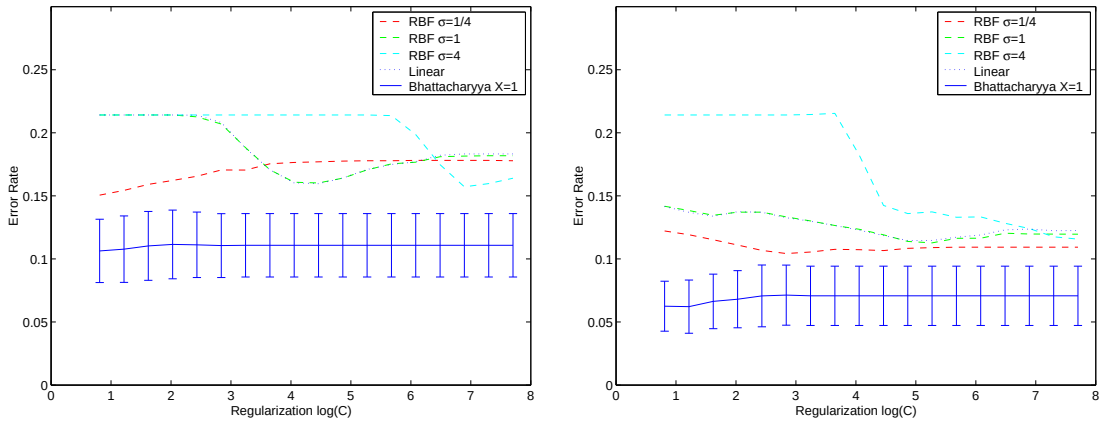
where α and β are user-defined scalar parameters. However, this kernel may not always satisfy Mercer's condition and a formal proof is not available. Another interesting connection is that this form is very similar to our mean-field bound on the probability product kernel (7). However, unlike the probability product kernel (and its bounds), computing the KL-divergence between complicated distributions (mixtures, hidden Markov models, Bayesian networks, linear dynamical systems and intractable graphical models) is intractable and must be approximated from the outset using numerical or sampling procedures which further decreases the possibility of having a valid Mercer kernel (Moreno et al., 2004).

8. Experiments

In this section we discuss three different learning problems: text classification, biological sequence classification and time series classification. For each problem, we select a generative model that is compatible with the domain which then leads to a specific form for the probability product kernel. For text, multinomial models are used, for sequences hidden Markov models are natural and for time series data we use linear dynamical systems. The subsequent kernel computations are fed to a discriminative classifier (a support vector machine) for training and testing.

8.1 Text Classification

For text classification, we used the standard WebKB data set which consists of HTML documents in multiple categories. Only the text component of each web page was preserved and HTML markup information and hyper-links were stripped away. No further stemming or elaborate text pre-processing was performed on the text. Subsequently, a bag-of-words model was used where each document is only represented by the frequency of words that appear within it. This corresponds to a multinomial distribution over counts. The frequencies of words are the maximum likelihood estimate for the multinomial generative model to a particular document which produces the vector of parameters $\hat{\alpha}$ as in Section 3.



(a) Training Set Size = 77 (b) Training Set Size = 622

Figure 7: SVM error rates (and standard deviation) for the probability product kernel (set at $\rho = 1/2$ as a Bhattacharyya kernel) for multinomial models as well as error rates for traditional kernels on the WebKB data set. Performance for multiple settings of the regularization parameter C in the support vector machine are shown. Two different sizes of training sets are shown (77 and 622). All results are shown for 20-fold cross-validation.

A support vector machine (which is known to have strong empirical performance on such text classification data) was then used to build a classifier. The SVM was fed our kernel's evaluations via a gram matrix and was trained to discriminate between faculty and student web pages in the WebKB data set (other categories were omitted). The probability product kernel was computed for multinomials under the parameter settings $\rho = 1/2$ (Bhattacharyya kernel) and $s = 1$ as in Section 3. Comparisons were made with the (linear) dot product kernel as well as Gaussian RBF kernels. The data set contains a total of 1641 student web pages and 1124 faculty web pages. The data for each class is further split into 4 universities and 1 miscellaneous category and we performed the usual training and testing split as described by (Lafferty and Lebanon, 2002; Joachims et al., 2001) where testing is performed on a held out university. The average error was computed from 20-fold cross-validation for the different kernels as a function of the support vector machine regularization parameter C in Figure 7. The figure shows error rates for different sizes of the training set (77 and 622 training points). In addition, we show the standard deviation of the error rate for the Bhattacharyya kernel. Even though we only explored a single setting of $s = 1, \rho = 1/2$ for the probability

product kernel, it outperforms the linear kernel as well as the RBF kernel at multiple settings of the RBF σ parameter (we attempted $\sigma = \{1/4, 1, 4\}$). This result confirms previous intuitions in the text classification community which suggest using squashing functions on word frequencies such as the logarithm or the square root (Goldzmidt and Sahami, 1998; Cutting et al., 1992). This also related to the power-transformation used in statistics known as the Box-Cox transformation which may help make data seem more Gaussian (Davidson and MacKinnon, 1993).

8.2 Biological Sequence Classification

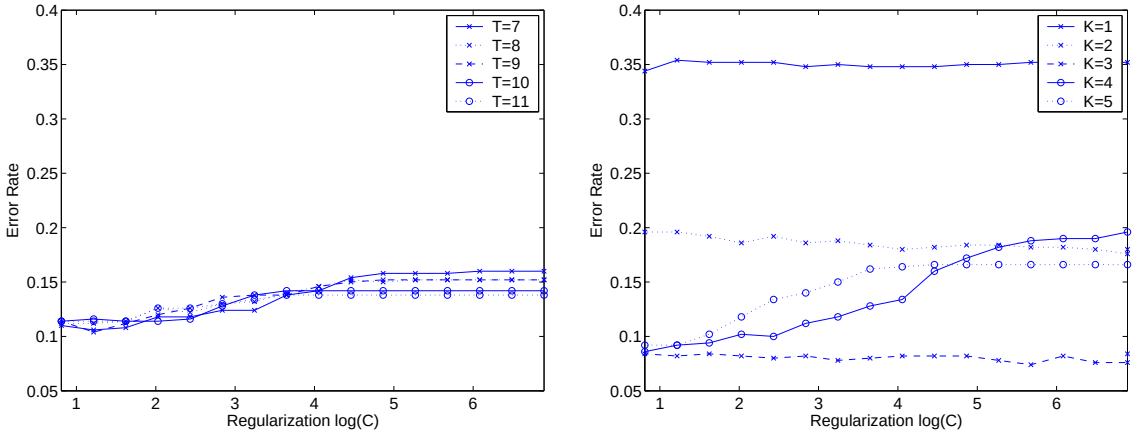
For discrete sequences, natural generative models to consider are Markov models and hidden Markov models. We obtained labeled gene sequences from the HS^3D data set[2] The sequences are of variable lengths and have discrete symbol entries from the 4-letter alphabet (G,A,T,C). We built classifiers to distinguish between gene sequences of two different types: introns and exons using raw sequences of varying lengths (from dozens of characters to tens of thousands of characters). From the original (unwindowed) 4450 introns and 3752 exons extracted directly from GenBank, we selected a random subset of 500 introns and 500 exons. These 1000 sequences were then randomly split into a 50% training and a 50% testing subset. In the first experiment we used the training data to learn stationary hidden Markov models whose number of states M was related to the length T_n of a given sequence n as follows: $M = \text{floor}(\frac{1}{2}\sqrt{O^2 + 4(T\zeta + O + 1)} - \frac{1}{2}O) + 1$. Here, O is the number of possible emission symbols (for gene sequences, $O = 4$) and ζ is the ratio of parameters to the number of symbols in the sequence T (we used $\zeta = 1/10$ although higher values may help avoid over-parameterizing the models). Each hidden Markov model was built from each sequence using the standard Expectation-Maximization algorithm (which is iterated for 400 steps to ensure convergence, this is particularly crucial for longer sequences). Gram matrices of size 1000×1000 were then formed by computing the probability product kernel between all the hidden Markov models. It was noted that all Gram matrices were positive definite by a singular value decomposition. We also used the following standard normalization of the kernel (a typical pre-processing step in the literature):

$$\tilde{k}(p, p') \quad \leftarrow \quad \frac{\tilde{k}(p, p')}{\sqrt{\tilde{k}(p, p)}\sqrt{\tilde{k}(p', p')}}.$$

Subsequently, we trained a support vector machine classifier on 500 example sequences and tested on the remaining 500. Figure 8(a) depicts the resulting error rate as we vary C, the regularization parameter on the SVM as well as T the number of time steps used by the kernel. Throughout, these experiments, we only used the setting $\rho = 1$. Note that these models were 1st order hidden Markov models where each state only depends on the previous state. Varying T, the length of the hidden Markov models used in the kernel computation has a slight effect on performance and it seems $T = 9$ at an appropriate C value performed best with an error rate as low as 10-11%.

For comparison, we also computed more standard string kernels such as the k-mer counts or spectrum kernels on the same training and testing gene sequences as shown in Figure 8(b). These basically correspond to a fully observed (non-hidden) stationary Markov model as in Figure 9. The zeroth order ($K = 1$) Markov model does poorly with its lowest error rate around 34%. The first

2. This is a collection of unprocessed intron and exon class gene sequences referred to as the *homo sapiens splice sites data set* and was downloaded from www.sci.unisannio.it/docenti/rampone.



(a) Hidden Markov Models (1st Order) (b) Markov Models (0th to 4th Order)

Figure 8: SVM error rates for the probability product kernel at $\rho = 1$ for hidden Markov models and Markov models (equivalent to string or spectrum kernels). In (a) test error rate under various levels of regularization is shown for 5 different settings of T in the probability product kernel. In (b) test error rate under various levels of regularization is shown for 5 different settings of the order K of the Markov model.

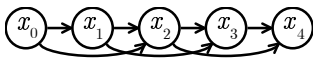


Figure 9: A higher order (2nd order or $K = 3$) Markov model.

order Markov model ($K = 2$) performs better with its lowest error rate at 18% yet is slightly worse than first order hidden Markov models. This helps motivate the possibility that dependence on the past in first order Markov models could be better modeled by a latent state as opposed to just the previous emission. This is despite the maximum likelihood estimation issues and local minima that we must contend with when training hidden Markov models using expectation-maximization. The second order Markov models with $K = 3$ fare best with an error rate of about 7-8%. However, higher orders ($K = 4$ and $K = 5$) seem to reduce performance. Therefore, a natural next experiment would be to consider higher order hidden Markov models, most notably second order ones where the emission distributions are conditioned on the past two states as opposed to the past two emitted symbols in the sequence.

8.3 Time Series Experiments

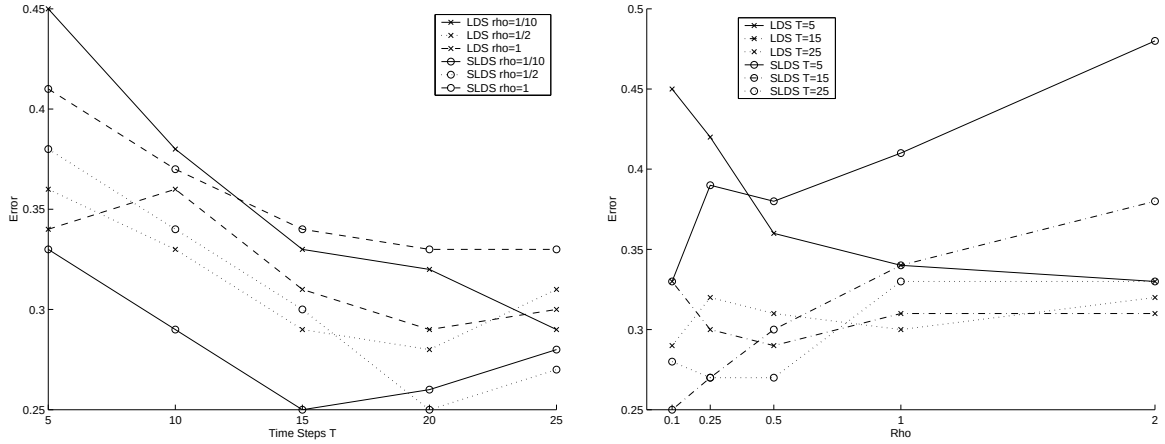
In this experiment we compare the performance of the probability product kernel between LDSs and the sampling approximation of the SLDS kernel with more traditional kernels and maximum likelihood techniques on synthetic data. We sampled from 20 exemplar distributions to generate synthetic time series data. The sampled distributions were 5 state, 2 dimensional SLDS models generated at random. Each model was sampled 5 times for 25 time steps with 10 models being assigned to each class. This gave 50 time series of length 25 per class, creating a challenging classification problem. For comparison, maximum likelihood models for LDSs and SLDSs were fit to the data and used for classification as well as SVMs trained using Gaussian RBF kernels. All testing was performed using leave one out cross validation. The SLDS kernel was approximated using sampling (50 samples) and the probability product kernels were normalized to improve performance.

The maximum likelihood LDS had an error rate of .35, the SLDS .30, and the Gaussian RBF .34. Exploring the setting of the parameters ρ and T for the LDS and SLDS kernels, we were able to outperform the maximum likelihood methods and the Gaussian RBF kernel with an optimal error for the LDS and SLDS kernel of .28 and .25 respectively. Figure 10 (a) and (b) show the error rate versus T and ρ respectively. It can be seen that increasing T generally improves performance which is to be expected. Although it does appear that T can be chosen too large. Meanwhile, ρ, generally appears to perform better when it is smaller (a counter example is the LDS kernel at the setting $T = 5$). Overall, the kernels performed comparably to or better than standard methods for most settings of ρ and T with the exception of extreme settings.

9. Conclusions

Discriminative learning, statistical learning theory and kernel-based algorithms have brought mathematical rigor and practical success to the field making it is easy to overlook the advantages of generative probabilistic modeling. Yet generative models are intuitive and offer flexibility for inserting structure, handling unusual input spaces, and accommodating prior knowledge. By proposing a kernel between the models themselves, this paper provides a bridge between the two schools of thought.

The form of the kernel is almost as simple as possible, yet it still gives rise to interesting nonlinearities and properties when applied to different generative models. It can be computed explicitly for some of the most commonly used parametric distributions in statistics such as the exponential family. For more elaborate models (graphical models, hidden Markov models, linear dynamical systems, etc.), computing the kernel reduces to using standard algorithms in the field. Experiments



(a) (b)

Figure 10: A comparison of the choice of parameters (a) T and (b) ρ. The x-axis is the parameter and the y-axis is the error rate.

show that probability product kernels hold promise in practical domains. Ultimately, engineering kernels between structured, discrete or unusual objects is an area of active research and, via generative modeling, probability product kernels can bring additional flexibility, formalism and potential.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. This work was funded in part by the National Science Foundation (grants IIS-0347499 and CCR-0312690).

References

Y. Bengio and P. Frasconi. Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, September 1996.

A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math Soc.*, 1943.

M. Collins and N. Duffy. Convolution kernels for natural language. In *Neural Information Processing Systems 14*, 2002.

C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In *Neural Information Processing Systems 15*, 2002.

D. R. Cutting, D. R. Karger, J. O. Pederson, and KJ. W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings ACM/SIGIR*, 1992.

R. Davidson and J. MacKinnon. *Estimation and Inference in Econometrics.* Oxford University Press, 1993.

Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29:245–273, 1997.

M. Goldzmidt and M. Sahami. A probabilistic approach to full-text document clustering. Technical report, Stanford University, 1998. Database Group Publication 60.

T. Hastie, R. Tibshirani, and Friedman J. H. *The Elements of Statistical Learning*. Springer Verlag, 2001.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

T. Jaakkola. Tutorial on variational approximation methods. In *Advanced Mean Field Methods: Theory and Practice*. MIT Press, 2000.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Neural Information Processing Systems 11*, 1998.

T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *Neural Information Processing Systems 12*, 1999.

T. Jebara and R. Kondor. Bhattacharyya and expected likelihood kernels. In *Conference on Learning Theory*, 2003.

T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorisation. In *International Conference on Machine Learning*, 2001.

M. Jordan and C. Bishop. *Introduction to Graphical Models*. In progress, 2004.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Machine Learning: Tenth International Conference, ICML*, 2003.

R. Kondor and T. Jebara. A kernel between sets of vectors. In *Machine Learning: Tenth International Conference*, 2003.

J. Lafferty and G. Lebanon. Information diffusion kernels. In *Neural Information Processing Systems*, 2002.

C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In *Neural Information Processing Systems*, 2002.

P. J. Moreno, P. P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Neural Information Processing Systems*, 2004.

C. Ong, A. Smola, and R. Williamson. Superkernels. In *Neural Information Processing Systems*, 2002.

C. S. Ong, M. Xavier, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *ICML*, 2004.

V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *Neural Information Processing Systems 13*, pages 981–987, 2000.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1997.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *J. of Time Series Analysis*, 3(4):253–264, 1982.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall: London., 1986.

F. Topsoe. Some inequalities for information divergence and related measures of discrimination. *J. of Inequalities in Pure and Applied Mathematics*, 2(1), 1999.

K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18 (90001):S268–S275, 2002.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

S. V. N. Vishawanathan and A. J. Smola. Fast kernels for string and tree matching. In *Neural Information Processing Systems 15*, 2002.

C. Watkins. *Advances in kernel methods*, chapter Dynamic Alignment Kernels. MIT Press, 2000.

Journal of Machine Learning Research 5 (2004) 845–889 Submitted 11/2000; Published 8/04

Feature Selection for Unsupervised Learning

Jennifer G. Dy JDY@ECE.NEU.EDU
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115, USA

Carla E. Brodley BRODLEY@ECN.PURDUE.EDU
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907, USA

Editor: Stefan Wrobel

Abstract

In this paper, we identify two issues involved in developing an automated feature subset selection algorithm for unlabeled data: the need for finding the number of clusters in conjunction with feature selection, and the need for normalizing the bias of feature selection criteria with respect to dimension. We explore the feature selection problem and these issues through FSSEM (Feature Subset Selection using Expectation-Maximization (EM) clustering) and through two different performance criteria for evaluating candidate feature subsets: scatter separability and maximum likelihood. We present proofs on the dimensionality biases of these feature criteria, and present a cross-projection normalization scheme that can be applied to any criterion to ameliorate these biases. Our experiments show the need for feature selection, the need for addressing these two issues, and the effectiveness of our proposed solutions.

Keywords: clustering, feature selection, unsupervised learning, expectation-maximization

1. Introduction

In this paper, we explore the issues involved in developing automated feature subset selection algorithms for unsupervised learning. By unsupervised learning we mean unsupervised classification, or clustering. Cluster analysis is the process of finding "natural" groupings by grouping "similar" (based on some similarity measure) objects together.

For many learning domains, a human defines the features that are potentially useful. However, not all of these features may be relevant. In such a case, choosing a subset of the original features will often lead to better performance. Feature selection is popular in supervised learning (Fukunaga, 1990; Almuallim and Dietterich, 1991; Cardie, 1993; Kohavi and John, 1997). For supervised learning, feature selection algorithms maximize some function of predictive accuracy. Because we are given class labels, it is natural that we want to keep only the features that are related to or lead to these classes. But in unsupervised learning, we are not given class labels. Which features should we keep? Why not use all the information we have? The problem is that not all features are important. Some of the features may be redundant, some may be irrelevant, and some can even misguide clustering results. In addition, reducing the number of features increases comprehensibility and ameliorates the problem that some unsupervised learning algorithms break down with high dimensional data.

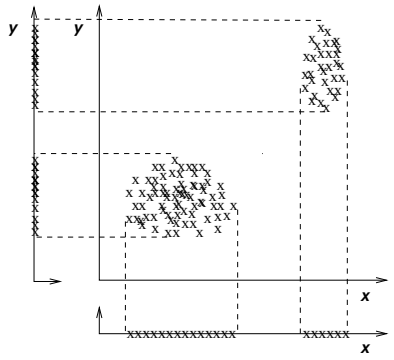


Figure 1: In this example, features x and y are redundant, because feature x provides the same information as feature y with regard to discriminating the two clusters.

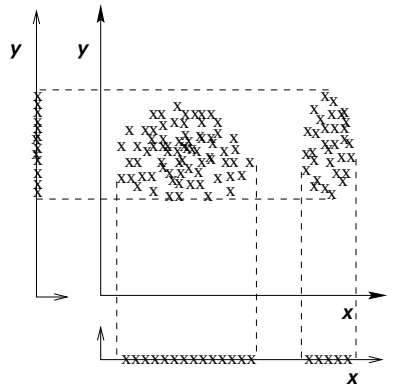
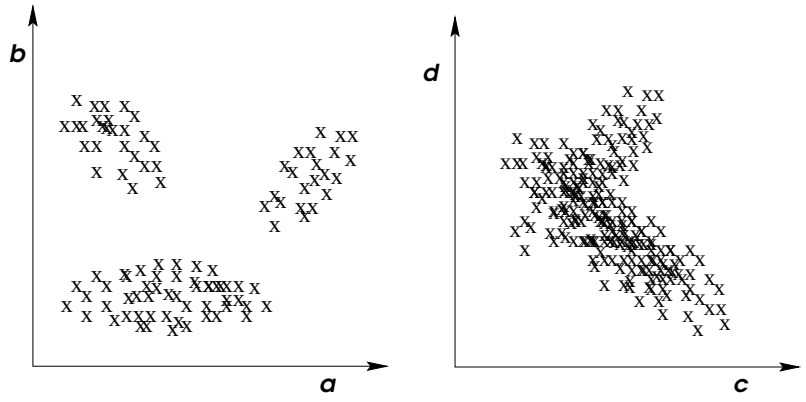


Figure 2: In this example, we consider feature y to be irrelevant, because if we omit x, we have only one cluster, which is uninteresting.

Figure 1 shows an example of feature redundancy for unsupervised learning. Note that the data can be grouped in the same way using only either feature x or feature y. Therefore, we consider features x and y to be redundant. Figure 2 shows an example of an irrelevant feature. Observe that feature y does not contribute to cluster discrimination. Used by itself, feature y leads to a single cluster structure which is uninteresting. Note that irrelevant features can misguide clustering results (especially when there are more irrelevant features than relevant ones). In addition, the situation in unsupervised learning can be more complex than what we depict in Figures 1 and 2. For example, in Figures 3a and b we show the clusters obtained using the feature subsets: $\{a,b\}$ and $\{c,d\}$ respectively. Different feature subsets lead to varying cluster structures. Which feature set should we pick?

Unsupervised learning is a difficult problem. It is more difficult when we have to simultaneously find the relevant features as well. A key element to the solution of any problem is to be able to precisely define the problem. In this paper, we define our task as:



(a) (b)

Figure 3: A more complex example. Figure a is the scatterplot of the data on features *a* and *b*. Figure b is the scatterplot of the data on features *c* and *d*.

> *The goal of feature selection for unsupervised learning is to find the smallest feature subset that best uncovers "interesting natural" groupings (clusters) from data according to the chosen criterion.*

There may exist multiple redundant feature subset solutions. We are satisfied in finding any one of these solutions. Unlike supervised learning, which has class labels to guide the feature search, in unsupervised learning we need to define what "interesting" and "natural" mean. These are usually represented in the form of criterion functions. We present examples of different criteria in Section 2.3.

Since research in feature selection for unsupervised learning is relatively recent, we hope that this paper will serve as a guide to future researchers. With this aim, we

1. Explore the wrapper framework for unsupervised learning,
2. Identify the issues involved in developing a feature selection algorithm for unsupervised learning within this framework,
3. Suggest ways to tackle these issues,
4. Point out the lessons learned from this endeavor, and
5. Suggest avenues for future research.

The idea behind the wrapper approach is to cluster the data as best we can in each candidate feature subspace according to what "natural" means, and select the most "interesting" subspace with the minimum number of features. This framework is inspired by the supervised wrapper approach (Kohavi and John, 1997), but rather than wrap the search for the best feature subset around a supervised induction algorithm, we wrap the search around a clustering algorithm.

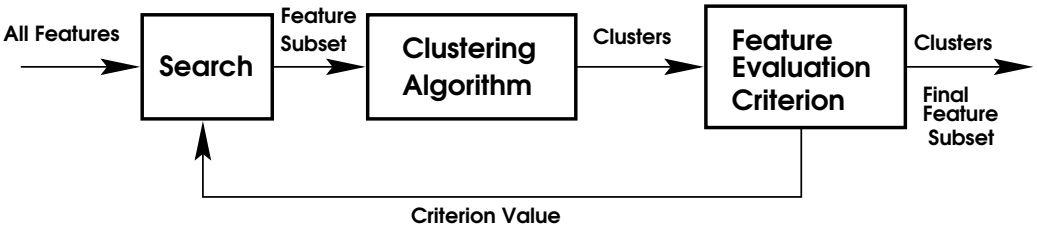


Figure 4: Wrapper approach for unsupervised learning.

In particular, this paper investigates the wrapper framework through FSSEM (feature subset selection using EM clustering) introduced in (Dy and Brodley, 2000a). Here, the term "EM clustering" refers to the expectation-maximization (EM) algorithm (Dempster et al., 1977; McLachlan and Krishnan, 1997; Moon, 1996; Wolfe, 1970; Wu, 1983) applied to estimating the maximum likelihood parameters of a finite Gaussian mixture. Although we apply the wrapper approach to EM clustering, the framework presented in this paper can be applied to any clustering method. FSSEM serves as an example. We present this paper such that applying a different clustering algorithm or feature selection criteria would only require replacing the corresponding clustering or feature criterion.

In Section 2, we describe FSSEM. In particular, we present the search method, the clustering method, and the two different criteria we selected to guide the feature subset search: scatter separability and maximum likelihood. By exploring the problem in the wrapper framework, we encounter and tackle two issues:

1. different feature subsets have different numbers of clusters, and

2. the feature selection criteria have biases with respect to feature subset dimensionality.

In Section 3, we discuss the complications that finding the number of clusters brings to the simultaneous feature selection/clustering problem and present one solution (FSSEM-k). Section 4 presents a theoretical explanation of why the feature selection criterion biases occur, and Section 5 provides a general normalization scheme which can ameliorate the biases of any feature criterion toward dimension.

Section 6 presents empirical results on both synthetic and real-world data sets designed to answer the following questions: (1) Is our feature selection for unsupervised learning algorithm better than clustering on all features? (2) Is using a fixed number of clusters, k, better than using a variable k in feature search? (3) Does our normalization scheme work? and (4) Which feature selection criterion is better? Section 7 provides a survey of existing feature selection algorithms. Section 8 provides a summary of the lessons learned from this endeavor. Finally, in Section 9, we suggest avenues for future research.

2. Feature Subset Selection and EM Clustering (FSSEM)

Feature selection algorithms can be categorized as either filter or wrapper (John et al., 1994) approaches. The filter approach basically pre-selects the features, and then applies the selected feature subset to the clustering algorithm. Whereas, the wrapper approach incorporates the clustering algorithm in the feature search and selection. We choose to explore the problem in the wrapper frame-

work because we are interested in understanding the interaction between the clustering algorithm and the feature subset search.

Figure 4 illustrates the wrapper approach. Our input is the set of all features. The output is the selected features and the clusters found in this feature subspace. The basic idea is to search through feature subset space, evaluating each candidate subset, F_t, by first clustering in space F_t using the clustering algorithm and then evaluating the resulting clusters and feature subset using our chosen feature selection criterion. We repeat this process until we find the best feature subset with its corresponding clusters based on our feature evaluation criterion. The wrapper approach divides the task into three components: (1) feature search, (2) clustering algorithm, and (3) feature subset evaluation.

2.1 Feature Search

An exhaustive search of the 2^d possible feature subsets (where d is the number of available features) for the subset that maximizes our selection criterion is computationally intractable. Therefore, a greedy search such as sequential forward or backward elimination (Fukunaga, 1990; Kohavi and John, 1997) is typically used. Sequential searches result in an $O(d^2)$ worst case search. In the experiments reported, we applied sequential forward search. Sequential forward search (SFS) starts with zero features and sequentially adds one feature at a time. The feature added is the one that provides the largest criterion value when used in combination with the features chosen. The search stops when adding more features does not improve our chosen feature criterion. SFS is not the best search method, nor does it guarantee an optimal solution. However, SFS is popular because it is simple, fast and provides a reasonable solution. For the purposes of our investigation in this paper, SFS would suffice. One may wish to explore other search methods for their wrapper approach. For example, Kim et al. (2002) applied evolutionary methods. Kittler (1978), and Russell and Norvig (1995) provide good overviews of different search strategies.

2.2 Clustering Algorithm

We choose EM clustering as our clustering algorithm, but other clustering methods can also be used in this framework. Recall that to cluster data, we need to make assumptions and define what "natural" grouping means. We apply the standard assumption that each of our "natural" groups is Gaussian. This assumption is not too limiting because we allow the number of clusters to adjust to our data, i.e., aside from finding the clusters we also find the number of "Gaussian" clusters. In Section 3, we discuss and present a solution to finding the number of clusters in conjunction with feature selection. We provide a brief description of EM clustering (the application of EM to approximate the maximum likelihood estimate of a finite mixture of multivariate Gaussians) in Appendix A. One can obtain a detailed description of EM clustering in (Fraley and Raftery, 2000; McLachlan and Krishnan, 1997). The Gaussian mixture assumption limits the data to continuous valued attributes. However, the wrapper framework can be extended to other mixture probability distributions (McLachlan and Basford, 1988; Titterington et al., 1985) and to other clustering methods, including graph theoretic approaches (Duda et al., 2001; Fukunaga, 1990; Jain and Dubes, 1988).

2.3 Feature Subset Selection Criteria

In this section, we investigate the feature subset evaluation criteria. Here, we define what "interestingness" means. There are two general views on this issue. One is that the criteria defining "interestingness" (feature subset selection criteria) should be the criteria used for clustering. The other is that the two criteria need not be the same. Using the same criteria for both clustering and feature selection provides a consistent theoretical optimization formulation. Using two different criteria, on the other hand, presents a natural way of combining two criteria for checks and balances. Proof on which view is better is outside the scope of this paper and is an interesting topic for future research. In this paper, we look at two feature selection criteria (one similar to our clustering criterion and the other with a different bias).

Recall that our goal is to find the feature subset that best discovers "interesting" groupings from data. To select an optimal feature subset, we need a measure to assess cluster quality. The choice of performance criterion is best made by considering the goals of the domain. In studies of performance criteria a common conclusion is: "Different classifications [clusterings] are right for different purposes, so we cannot say any one classification is best." – Hartigan, 1985 .

In this paper, we do not attempt to determine the best criterion (one can refer to Milligan (1981) on comparative studies of different clustering criteria). We investigate two well-known measures: scatter separability and maximum likelihood. In this section, we describe each criterion, emphasizing the assumptions made by each.

Scatter Separability Criterion: A property typically desired among groupings is cluster separation. We investigate the scatter matrices and separability criteria used in discriminant analysis (Fukunaga, 1990) as our feature selection criterion. We choose to explore the scatter separability criterion, because it can be used with any clustering method.[1] The criteria used in discriminant analysis assume that the features we are interested in are features that can group the data into clusters that are unimodal and separable.

S_w is the within-class scatter matrix and S_b is the between class scatter matrix, and they are defined as follows:

$$S_w = \sum_{j=1}^{k} \pi_j E\{(X-\mu_j)(X-\mu_j)^T|\omega_j\} = \sum_{j=1}^{k} \pi_j \Sigma_j, \quad (1)$$

$$S_b = \sum_{j=1}^{k} \pi_j (\mu_j - M_o)(\mu_j - M_o)^T, \quad (2)$$

$$M_o = E\{X\} = \sum_{j=1}^{k} \pi_j \mu_j, \quad (3)$$

where π_j is the probability that an instance belongs to cluster ω_j, X is a d-dimensional random feature vector representing the data, k the number of clusters, μ_j is the sample mean vector of cluster ω_j, M_o is the total sample mean, Σ_j is the sample covariance matrix of cluster ω_j, and $E\{\cdot\}$ is the expected value operator.

1. One can choose to use the non-parametric version of this criterion measure (Fukunaga, 1990) for non-parametric clustering algorithms.

S_w measures how scattered the samples are from their cluster means. S_b measures how scattered the cluster means are from the total mean. We would like the distance between each pair of samples in a particular cluster to be as small as possible and the cluster means to be as far apart as possible with respect to the chosen similarity metric (Euclidean, in our case). Among the many possible separability criteria, we choose the $trace(S_w^{-1}S_b)$ criterion because it is invariant under any nonsingular linear transformation (Fukunaga, 1990). Transformation invariance means that once m features are chosen, any nonsingular linear transformation on these features does not change the criterion value. This implies that we can apply weights to our m features or apply any nonsingular linear transformation or projection to our features and still obtain the same criterion value. This makes the $trace(S_w^{-1}S_b)$ criterion more robust than other variants. $S_w^{-1}S_b$ is S_b normalized by the average cluster covariance. Hence, the larger the value of $trace(S_w^{-1}S_b)$ is, the larger the normalized distance between clusters is, which results in better cluster discrimination.

Maximum Likelihood (ML) Criterion: By choosing EM clustering, we assume that each grouping or cluster is Gaussian. We maximize the likelihood of our data given the parameters and our model. Thus, maximum likelihood (ML) tells us how well our model, here a Gaussian mixture, fits the data. Because our clustering criterion is ML, a natural criterion for feature selection is also ML. In this case, the "interesting" groupings are the "natural" groupings, i.e., groupings that are Gaussian.

3. The Need for Finding the Number of Clusters (FSSEM-k)

When we are searching for the best subset of features, we run into a new problem: that the number of clusters, *k, depends on the feature subset*. Figure 5 illustrates this point. In two dimensions (shown on the left) there are three clusters, whereas in one-dimension (shown on the right) there are only two clusters. Using a fixed number of clusters for all feature sets does not model the data in the respective subspace correctly.

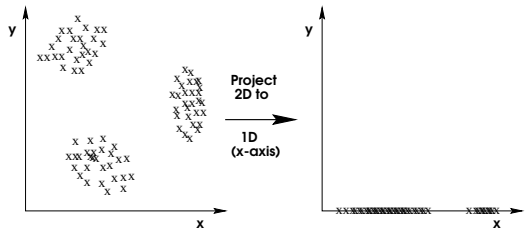


Figure 5: The number of cluster components varies with dimension.

Unsupervised clustering is made more difficult when we do not know the number of clusters, k. To search for k for a given feature subset, FSSEM-k currently applies Bouman et al.'s method (1998) for merging clusters and adds a Bayesian Information Criterion (BIC) (Schwarz, 1978) penalty term to the log-likelihood criterion. A penalty term is needed because the maximum likelihood estimate increases as more clusters are used. We do not want to end up with the trivial result wherein each data point is considered as an individual cluster. Our new objective function becomes: $F(k,\Phi) = \log(f(X|\Phi)) - \frac{1}{2}L\log(N)$ where N is the number of data points, L is the number of free

parameters in Φ, and $\log(f(X|\Phi))$ is the log-likelihood of our observed data X given the parameters Φ. Note that L and Φ vary with k.

Using Bouman et al.'s method (1998), we begin our search for k with a large number of clusters, K_{max}, and then sequentially decrement this number by one until only one cluster remains (a merge method). Other methods start from $k=1$ and add more and more clusters as needed (split methods), or perform both split and merge operations (Ueda et al., 1999). To initialize the parameters of the $(k-1)$th model, two clusters from the kth model are merged. We choose the two clusters among all pairs of clusters in k, which when merged give the minimum difference between $F(k-1,\Phi)$ and $F(k,\Phi)$. The parameter values that are not merged retain their value for initialization of the $(k-1)$th model. The parameters for the merged cluster (l and m) are initialized as follows:

$$\begin{aligned}
\pi_j^{k-1,(0)} &= \pi_l + \pi_m; \\
\mu_j^{k-1,(0)} &= \tfrac{\pi_l\mu_l+\pi_m\mu_m}{\pi_l+\pi_m}; \\
\Sigma_j^{k-1,(0)} &= \tfrac{\pi_l(\Sigma_l+(\mu_l-\mu_j^{k-1,(0)})(\mu_l-\mu_j^{k-1,(0)})^T)+\pi_m(\Sigma_m+(\mu_m-\mu_j^{k-1,(0)})(\mu_m-\mu_j^{k-1,(0)})^T)}{\pi_l+\pi_m};
\end{aligned}$$

where the superscript $k-1$ indicates the $k-1$ cluster model and the superscript (0) indicates the first iteration in this reduced order model. For each candidate k, we iterate EM until the change in $F(k,\Phi)$ is less than ε (default 0.0001) or up to n (default 500) iterations. Our algorithm outputs the number of clusters k, the parameters, and the clustering assignments that maximize the $F(k,\Phi)$ criterion (our modified ML criterion).

There are myriad ways to find the "optimal" number of clusters k with EM clustering. These methods can be generally grouped into three categories: hypothesis testing methods (McLachlan and Basford, 1988), penalty methods like AIC (Akaike, 1974), BIC (Schwarz, 1978) and MDL (Rissanen, 1983), and Bayesian methods like AutoClass (Cheeseman and Stutz, 1996). Smyth (1996) introduced a new method called Monte Carlo cross-validation (MCCV). For each possible k value, the average cross-validated likelihood on M runs is computed. Then, the k value with the highest cross-validated likelihood is selected. In an experimental evaluation, Smyth showed that MCCV and AutoClass found k values that were closer to the number of classes than the k values found with BIC for their data sets. We chose Bouman et al.'s method with BIC, because MCCV is more computationally expensive. MCCV has complexity $O(MK_{max}^2d^2NE)$, where M is the number of cross-validation runs, K_{max} is the maximum number of clusters considered, d is the number of features, N is the number of samples and E is the average number of EM iterations. The complexity of Bouman et al.'s approach is $O(K_{max}^2d^2NE')$. Furthermore, for $k<K_{max}$, we do not need to re-initialize EM (because we merged two clusters from $k+1$) resulting in $E'<E$. Note that in FSSEM, we run EM for each candidate feature subset. Thus, in feature selection, the total complexity is the complexity of each complete EM run times the feature search space. Recently, Figueiredo and Jain (2002) presented an efficient algorithm which integrates estimation and model selection for finding the number of clusters using minimum message length (a penalty method). It would be of interest for future work to examine these other ways for finding k coupled with feature selection.

4. Bias of Criterion Values to Dimension

Both feature subset selection criteria have biases with respect to dimension. We need to analyze these biases because in feature subset selection we compare the criterion values for subsets of dif-

ferent cardinality (corresponding to different dimensionality). In Section 5, we present a solution to this problem.

4.1 Bias of the Scatter Separability Criterion

The separability criterion prefers higher dimensionality; i.e., the criterion value monotonically increases as features are added assuming identical clustering assignments (Fukunaga, 1990; Narendra and Fukunaga, 1977). However, the separability criterion may not be monotonically increasing with respect to dimension when the clustering assignments change.

Scatter separability or the *trace* criterion prefers higher dimensions, intuitively, because data is more scattered in higher dimensions, and mathematically, because more features mean adding more terms in the *trace* function. Observe that in Figure 6, feature y does not provide additional discrimination to the two-cluster data set. Yet, the *trace* criterion prefers feature subset $\{x,y\}$ over feature subset $\{x\}$. Ideally, we would like the criterion value to remain the same if the discrimination information is the same.

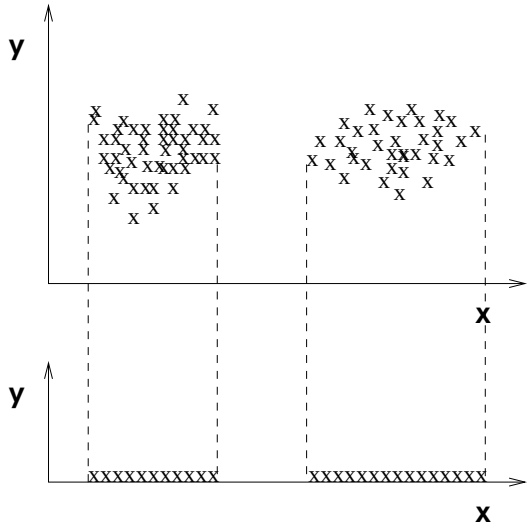


Figure 6: An illustration of scatter separability's bias with dimension.

The following simple example provides us with an intuitive understanding of this bias. Assume that feature subset S_1 and feature subset S_2 produce identical clustering assignments, $S_1 \subset S_2$ where S_1 and S_2 have d and $d+1$ features respectively. Assume also that the features are uncorrelated within each cluster. Let S_{w_d} and S_{b_d} be the within-class scatter and between-class scatter in dimension d respectively. To compute $trace(S_{w_{d+1}}^{-1} S_{b_{d+1}})$ for $d+1$ dimensions, we simply add a positive term to the $trace(S_{w_d}^{-1} S_{b_d})$ value for d dimensions. $S_{w_{d+1}}$ and $S_{b_{d+1}}$ in the $d+1$ dimensional space are computed as

$$S_{w_{d+1}} = \begin{bmatrix} S_{w_d} & 0 \\ 0 & \sigma^2_{w_{d+1}} \end{bmatrix}$$

and

$$S_{b_{d+1}} = \begin{bmatrix} S_{b_d} & 0 \\ 0 & \sigma^2_{b_{d+1}} \end{bmatrix}.$$

Since

$$S_{w_{d+1}}^{-1} = \begin{bmatrix} S_{w_d}^{-1} & 0 \\ 0 & \frac{1}{\sigma_{w_{d+1}}^2} \end{bmatrix},$$

$trace(S_{w_{d+1}}^{-1} S_{b_{d+1}})$ would be $trace(S_{w_d}^{-1} S_{b_d}) + \frac{\sigma_{b_{d+1}}^2}{\sigma_{w_{d+1}}^2}$. Since $\sigma_{b_{d+1}}^2 \geq 0$ and $\sigma_{w_{d+1}}^2 > 0$, the *trace* of the $d+1$ clustering will always be greater than or equal to *trace* of the d clustering under the stated assumptions.

The separability criterion monotonically increases with dimension even when the features are correlated as long as the clustering assignments remain the same. Narendra and Fukunaga (1977) proved that a criterion of the form $X_d^T S_d^{-1} X_d$, where X_d is a d-column vector and S_d is a $d \times d$ positive definite matrix, monotonically increases with dimension. They showed that

$$X_{d-1}^T S_{d-1}^{-1} X_{d-1} = X_d^T S_d^{-1} X_d - \frac{1}{b}[(C^T : b) X_d]^2, \tag{4}$$

where

$$X_d = \begin{bmatrix} X_{d-1} \\ x_d \end{bmatrix},$$

$$S_d^{-1} = \begin{bmatrix} A & C \\ C^T & b \end{bmatrix},$$

X_{d-1} and C are $d-1$ column vectors, x_d and b are scalars, A is a $(d-1) \times (d-1)$ matrix, and the symbol : means matrix augmentation. We can show that $trace(S_{w_d}^{-1} S_{b_d})$ can be expressed as a criterion of the form $\sum_{j=1}^{k} X_{jd}^T S_d^{-1} X_{jd}$. S_{b_d} can be expressed as $\sum_{j=1}^{k} Z_{jb_d} Z_{jb_d}^T$ where Z_{jb_d} is a d-column vector:

$$\begin{aligned} trace(S_{w_d}^{-1} S_{b_d}) &= trace(S_{w_d}^{-1} \sum_{j=1}^{k} Z_{jb_d} Z_{jb_d}^T) \\ &= trace(\sum_{j=1}^{k} S_{w_d}^{-1} Z_{jb_d} Z_{jb_d}^T) \\ &= \sum_{j=1}^{k} trace(S_{w_d}^{-1} Z_{jb_d} Z_{jb_d}^T) \\ &= \sum_{j=1}^{k} trace(Z_{jb_d}^T S_{w_d}^{-1} Z_{jb_d}), \end{aligned}$$

since $trace(A_{p\times q} B_{q\times p}) = trace(B_{q\times p} A_{p\times q})$ for any rectangular matrices $A_{p\times q}$ and $B_{q\times p}$.

Because $Z_{jb_d}^T S_{w_d}^{-1} Z_{jb_d}$ is scalar,

$$\sum_{j=1}^{k} trace(Z_{jb_d}^T S_{w_d}^{-1} Z_{jb_d}) = \sum_{j=1}^{k} Z_{jb_d}^T S_{w_d}^{-1} Z_{jb_d}.$$

Since each term monotonically increases with dimension, the summation also monotonically increases with dimension. Thus, the scatter separability criterion increases with dimension assuming the clustering assignments remain the same. This means that even if the new feature does not facilitate finding new clusters, the criterion function increases.

4.2 Bias of the Maximum Likelihood (ML) Criterion

Contrary to finding the number of clusters problem, wherein ML increases as the number of model parameters (k) is increased, in feature subset selection, ML prefers lower dimensions. In finding the number of clusters, we try to fit the best Gaussian mixture to the data. The data is fixed and we try to fit our model as best as we can. In feature selection, given different feature spaces, we select the feature subset that is best modeled by a Gaussian mixture.

This bias problem occurs because we define likelihood as the likelihood of the data corresponding to the candidate feature subset (see Equation 10 in Appendix B). To avoid this bias, the comparison can be between two complete (relevant and irrelevant features included) models of the data. In this case, likelihood is defined such that the candidate relevant features are modeled as dependent on the clusters, and the irrelevant features are modeled as having no dependence on the cluster variable. The problem with this approach is the need to define a model for the irrelevant features. Vaithyanathan and Dom uses this for document clustering (Vaithyanathan and Dom, 1999). The multinomial distribution for the relevant and irrelevant features is an appropriate model for text features in document clustering. In other domains, defining models for the irrelevant features may be difficult. Moreover, modeling irrelevant features means more parameters to predict. This implies that we still work with all the features, and as we mentioned earlier, algorithms may break down with high dimensions; we may not have enough data to predict all model parameters. One may avoid this problem by adding the assumption of independence among irrelevant features which may not be true. A poorly-fitting irrelevant feature distribution may cause the algorithm to select too many features. Throughout this paper, we use the maximum likelihood definition only for the relevant features.

For a fixed number of samples, ML prefers lower dimensions. The problem occurs when we compare feature set A with feature set B wherein set A is a subset of set B, and the joint probability of a single point (x,y) is less than or equal to its marginal probability (x). For sequential searches, this can lead to the trivial result of selecting only a single feature.

ML prefers lower dimensions for discrete random features. The joint probability mass function of discrete random vectors X and Y is $p(X,Y) = p(Y|X)p(X)$. Since $0 \leq p(Y|X) \leq 1$, $p(X,Y) = p(Y|X)p(X) \leq p(X)$. Thus, $p(X)$ is always greater than or equal to $p(X,Y)$ for any X. When we deal with continuous random variables, as in this paper, the definition, $f(X,Y) = f(Y|X)f(X)$ still holds, where $f(\cdot)$ is now the probability density function. $f(Y|X)$ is always greater than or equal to zero. However, $f(Y|X)$ can be greater than one. The marginal density $f(X)$ is greater than or equal to the joint probability $f(X,Y)$ iff $f(Y|X) \leq 1$.

Theorem 4.1 *For a finite multivariate Gaussian mixture, assuming identical clustering assignments for feature subsets A and B with dimensions $d_B \geq d_A$, $ML(\Phi_A) \geq ML(\Phi_B)$ iff*

$$\prod_{j=1}^{k} \left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j} \right)^{\pi_j} \geq \frac{1}{(2\pi e)^{(d_B - d_A)}},$$

where Φ_A represents the parameters and Σ_{A_j} is the covariance matrix modelling cluster j in feature subset A, π_j is the mixture proportion of cluster j, and k is the number of clusters.

Corollary 4.1 *For a finite multivariate Gaussian mixture, assuming identical clustering assignments for feature subsets X and (X,Y), where X and Y are disjoint, $ML(\Phi_X) \geq ML(\Phi_{XY})$ iff*

$$\prod_{j=1}^{k} |\Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}|_j^{\pi_j} \quad \geq \quad \frac{1}{(2\pi e)^{d_Y}},$$

where the covariance matrix in feature subset (X,Y) is $\begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}$, and d_Y is the dimension in Y.

We prove Theorem 4.1 and Corollary 4.1 in Appendix B. Theorem 4.1 and Corollary 4.1 reveal the dependencies of comparing the ML criterion for different dimensions. Note that each jth component of the left hand side term of Corollary 4.1 is the determinant of the conditional covariance of $f(Y|X)$. This covariance term is the covariance of Y eliminating the effects of the conditioning variable X, i.e., the conditional covariance does not depend on X. The right hand side is approximately equal to $(0.06)^{d_Y}$. This means that the ML criterion increases when the feature or feature subset to be added (Y) has a generalized variance (determinant of the covariance matrix) smaller than $(0.06)^{d_Y}$. Ideally, we would like our criterion measure to remain the same when the subsets reveal the same clusters. Even when the feature subsets reveal the same cluster, Corollary 4.1 informs us that ML decreases or increases depending on whether or not the generalized variance of the new features is greater than or less than a constant respectively.

5. Normalizing the Criterion Values: Cross-Projection Method

The arguments from the previous section illustrate that to apply the ML and *trace* criteria to feature selection, we need to normalize their values with respect to dimension. A typical approach to normalization is to divide by a penalty factor. For example, for the scatter criterion, we could divide by the dimension, d. Similarly for the ML criterion, we could divide by $\frac{1}{(2\pi e)^d}$. But, $\frac{1}{(2\pi e)^d}$ would not remove the covariance terms due to the increase in dimension. We could also divide log ML by d, or divide only the portions of the criterion affected by d. The problem with dividing by a penalty is that it requires specification of a different magic function for each criterion.

The approach we take is to project our clusters to the subspaces that we are comparing. Given two feature subsets, S_1 and S_2, of different dimension, clustering our data using subset S_1 produces cluster C_1. In the same way, we obtain the clustering C_2 using the features in subset S_2. Which feature subset, S_1 or S_2, enables us to discover better clusters? Let $CRIT(S_i,C_j)$ be the feature selection criterion value using feature subset S_i to represent the data and C_j as the clustering assignment. $CRIT(\cdot)$ represents either of the criteria presented in Section 2.3. We normalize the criterion value for S_1, C_1 as

$$normalizedValue(S_1,C_1) = CRIT(S_1,C_1) \cdot CRIT(S_2,C_1),$$

and, the criterion value for S_2, C_2 as

$$normalizedValue(S_2,C_2) = CRIT(S_2,C_2) \cdot CRIT(S_1,C_2).$$

If $normalizedValue(S_i,C_i) > normalizedValue(S_j,C_j)$, we choose feature subset S_i. When the normalized criterion values are equal for S_i and S_j, we favor the lower dimensional feature subset. The

choice of a product or sum operation is arbitrary. Taking the product will be similar to obtaining the geometric mean, and a sum with an arithmetic mean. In general, one should perform normalization based on the semantics of the criterion function. For example, geometric mean would be appropriate for likelihood functions, and an arithmetic mean for the log-likelihood.

When the clustering assignments resulting from different feature subsets, S_1 and S_2, are identical (i.e., $C_1 = C_2$), the $normalizedValue(S_1, C_1)$ would be equal to the $normalizedValue(S_2, C_2)$, which is what we want. More formally:

Proposition 1 *Given that $C_1 = C_2$, equal clustering assignments, for two different feature subsets, S_1 and S_2, then $normalizedValue(S_1, C_1) = normalizedValue(S_2, C_2)$.*

Proof: From the definition of $normalizedValue(\cdot)$ we have

$$normalizedValue(S_1, C_1) \quad = \quad CRIT(S_1, C_1) \cdot CRIT(S_2, C_1).$$

Substituting $C_1 = C_2$,

$$\begin{aligned} normalizedValue(S_1, C_1) &= CRIT(S_1, C_2) \cdot CRIT(S_2, C_2). \\ &= normalizedValue(S_2, C_2). \end{aligned}$$ □

To understand why cross-projection normalization removes some of the bias introduced by the difference in dimension, we focus on $normalizedValue(S_1, C_1)$. The common factor is C_1 (the clusters found using feature subset S_1). We measure the criterion values on both feature subsets to evaluate the clusters C_1. Since the clusters are projected on both feature subsets, the bias due to data representation and dimension is diminished. The normalized value focuses on the quality of the clusters obtained.

For example, in Figure 7, we would like to see whether subset S_1 leads to better clusters than subset S_2. $CRIT(S_1, C_1)$ and $CRIT(S_2, C_2)$ give the criterion values of S_1 and S_2 for the clusters found in those feature subspaces (see Figures 7a and 7b). We project clustering C_1 to S_2 in Figure 7c and apply the criterion to obtain $CRIT(S_2, C_1)$. Similarly, we project C_2 to feature space S_1 to obtain the result shown in Figure 7d. We measure the result as $CRIT(S_1, C_2)$. For example, if $ML(S_1, C_1)$ is the maximum likelihood of the clusters found in subset S_1 (using Equation 10, Appendix B),[2] then to compute $ML(S_2, C_1)$, we use the same cluster assignments, C_1, i.e., the $E[z_{ij}]$'s (the membership probabilities) for each data point x_i remain the same. To compute $ML(S_2, C_1)$, we apply the maximization-step EM clustering update equations (Equations 7-9 in Appendix A to compute the model parameters in the increased feature space, $S_2 = \{F_2, F_3\}$.

Since we project data in both subsets, we are essentially comparing criteria in the same number of dimensions. We are comparing $CRIT(S_1, C_1)$ (Figure 7a) with $CRIT(S_1, C_2)$ (Figure 7d) and $CRIT(S_2, C_1)$ (Figure 7c) with $CRIT(S_2, C_2)$ (Figure 7b). In this example, normalized *trace* chooses subset S_2, because there exists a better cluster separation in both subspaces using C_2 rather than C_1. Normalized ML also chooses subset S_2. C_2 has a better Gaussian mixture fit (smaller variance clusters) in both subspaces (Figures 7b and d) than C_1 (Figures 7a and c). Note that the underlying

2. One can compute the maximum log-likelihood, log ML, efficiently as $Q(\Phi, \Phi) + H(\Phi, \Phi)$ by applying Lemma B.1 and Equation 16 in Appendix B. Lemma B.1 expresses the $Q(\cdot)$ in terms only of the parameter estimates. Equation 16, $H(\Phi, \Phi)$, is the cluster entropy which requires only the $E[z_{ij}]$ values. In practice, we work with log ML to avoid precision problems. The product $normalizedValue(\cdot)$ function then becomes $\log normalizedValue(S_i, C_i) = \log ML(S_i, C_i) + \log ML(S_j, C_i)$.

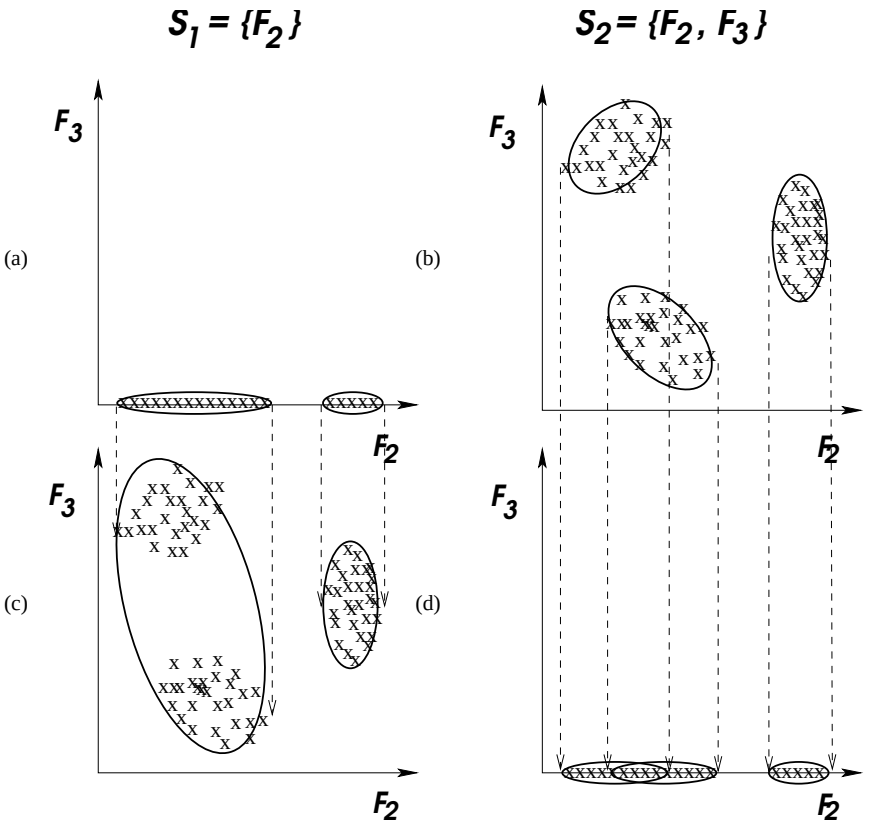


Figure 7: Illustration on normalizing the criterion values. To compare subsets, S_1 and S_2, we project the clustering results of S_1, we call C_1 in (a), to feature space S_2 as shown in (c). We also project the clustering results of S_2, C_2 in (b), onto feature space S_1 as shown in (d). In (a), $tr(S_1,C_1) = 6.094$, $ML(S_1,C_1) = 1.9 \times 10^{-64}$, and $\log ML(S_1,C_1) = -146.7$. In (b), $tr(S_2,C_2) = 9.390$, $ML(S_2,C_2) = 4.5 \times 10^{-122}$, and $\log ML(S_1,C_2) = -279.4$. In (c), $tr(S_2,C_1) = 6.853$, $ML(S_2,C_1) = 3.6 \times 10^{-147}$, and $\log ML(S_2,C_1) = -337.2$. In (d), $tr(S_1,C_2) = 7.358$, $ML(S_1,C_2) = 2.1 \times 10^{-64}$, and $\log ML(S_1,C_2) = -146.6$. We evaluate subset S_1 with *normalized* $tr(S_1,C_1) = 41.76$ and subset S_2 with *normalized* $tr(S_2,C_2) = 69.09$. In the same way, using ML, the normalized values are: 6.9×10^{-211} for subset S_1 and 9.4×10^{-186} for subset S_2. With log ML, the normalized values are: -483.9 and -426.0 for subsets S_1 and S_2 respectively.

assumption behind this normalization scheme is that the clusters found in the new feature space should be consistent with the structure of the data in the previous feature subset. For the ML criterion, this means that C_i should model S_1 and S_2 well. For the *trace* criterion, this means that the clusters C_i should be well separated in both S_1 and S_2.

6. Experimental Evaluation

In our experiments, we 1) investigate whether feature selection leads to better clusters than using all the features, 2) examine the results of feature selection with and without criterion normalization, 3) check whether or not finding the number of clusters helps feature selection, and 4) compare the ML and the *trace* criteria. We first present experiments with synthetic data and then a detailed analysis of the FSSEM variants using four real-world data sets. In this section, we first describe our synthetic Gaussian data, our evaluation methods for the synthetic data, and our EM clustering implementation details. We then present the results of our experiments on the synthetic data. Finally, in Section 6.5, we present and discuss experiments with three benchmark machine learning data sets and one new real world data set.

6.1 Synthetic Gaussian Mixture Data

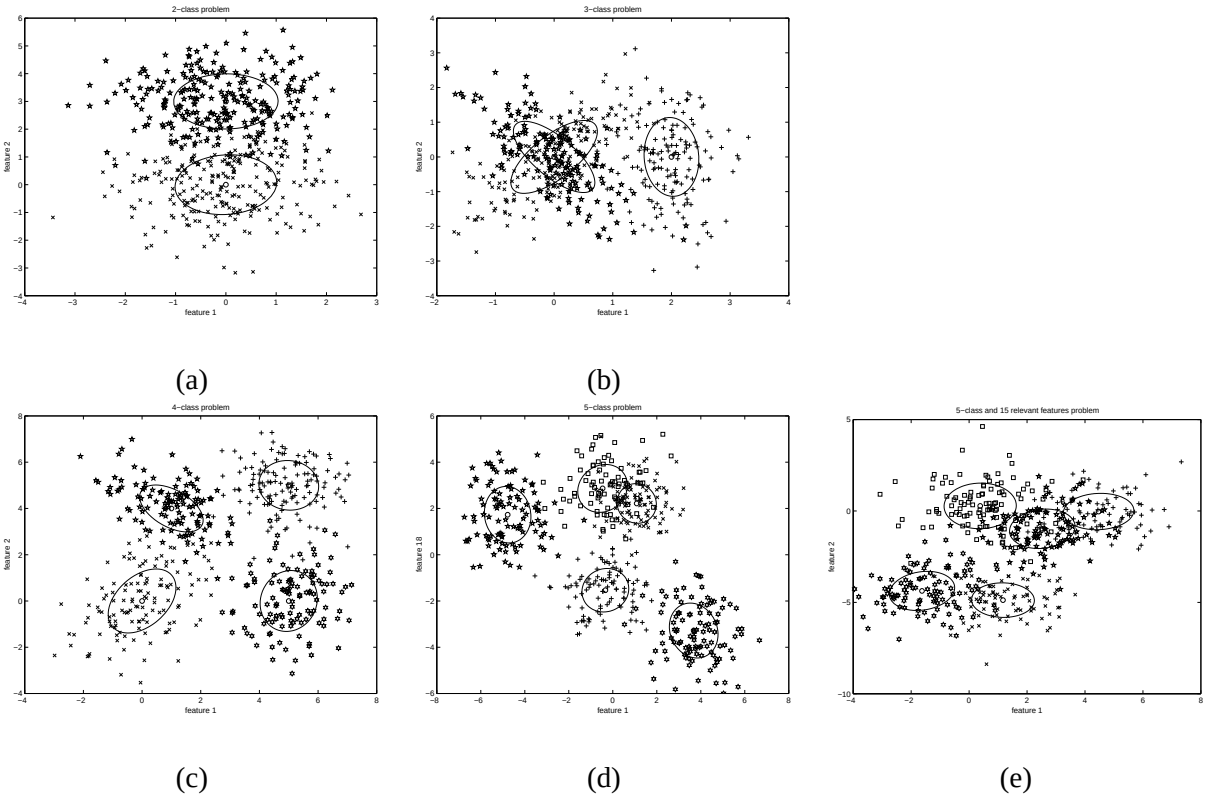


Figure 8: Synthetic Gaussian data.

To understand the performance of our algorithm, we experiment with five sets of synthetic Gaussian mixture data. For each data set we have "relevant" and "irrelevant" features, where relevant means that we created our k component mixture model using these features. Irrelevant features are generated as Gaussian normal random variables. For all five synthetic data sets, we generated $N = 500$ data points and generated clusters that are of equal proportions.

2-**class,** 2 **relevant features and** 3 **noise features:** The first data set (shown in Figure 8a) consists of two Gaussian clusters, both with covariance matrix, $\Sigma_1 = \Sigma_2 = I$ and means $\mu_1 = (0,0)$ and $\mu_2 = (0,3)$. This is similar to the two-class data set used by (Smyth, 1996). There is considerable overlap between the two clusters, and the three additional "noise" features increase the difficulty of the problem.

3-**class,** 2 **relevant features and** 3 **noise features:** The second data set consists of three Gaussian clusters and is shown in Figure 8b. Two clusters have means at $(0,0)$ but the covariance matrices are orthogonal to each other. The third cluster overlaps the tails on the right side of the other two clusters. We add three irrelevant features to the three-class data set used by (Smyth, 1996).

4-**class,** 2 **relevant features and** 3 **noise features:** The third data set (Figure 8c) has four clusters with means at $(0,0)$, $(1,4)$, $(5,5)$ and $(5,0)$ and covariances equal to I. We add three Gaussian normal random "noise" features.

5-**class,** 5 **relevant features and** 15 **noise features:** For the fourth data set, there are twenty features, but only five are relevant (features {1, 10, 18, 19, 20}). The true means μ were sampled from a uniform distribution on $[-5,5]$. The elements of the diagonal covariance matrices σ were sampled from a uniform distribution on $[0.7,1.5]$ (Fayyad et al., 1998). Figure 8d shows the scatter plot of the data in two of its relevant features.

5-**class,** 15 **relevant features and** 5 **noise features:** The fifth data set (Figure 8e shown in two of its relevant features) has twenty features with fifteen relevant features {1, 2, 3, 5, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 20}. The true means μ were sampled from a uniform distribution on $[-5,5]$. The elements of the diagonal covariance matrices σ were sampled from a uniform distribution on $[0.7,1.5]$ (Fayyad et al., 1998).

6.2 Evaluation Measures

We would like to measure our algorithm's ability to select relevant features, to correctly identify k, and to find structure in the data (clusters). There are no standard measures for evaluating clusters in the clustering literature (Jain and Dubes, 1988). Moreover, no single clustering assignment (or class label) explains every application (Hartigan, 1985). Nevertheless, we need some measure of performance. Fisher (1996) provides and discusses different internal and external criteria for measuring clustering performance.

Since we generated the synthetic data, we know the 'true' cluster to which each instance belongs. This 'true' cluster is the component that generates that instance. We refer to these 'true' clusters as our known 'class' labels. Although we used the class labels to measure the performance of FSSEM, we did not use this information during training (i.e., in selecting features and discovering clusters).

Cross-Validated Class Error: We define class error as the number of instances misclassified divided by the total number of instances. We assign each data point to its most likely cluster, and assign each cluster to a class based on examining the class labels of the training data assigned to each cluster and choosing the majority class. Since we have the true cluster labels, we can compute classification error. One should be careful when comparing clusterings with

different number of clusters using training error. Class error based on training decreases with an increase in the number of clusters, k, with the trivial result of 0% error when each data point is a cluster. To ameliorate this problem, we use ten-fold cross-validation error. Ten-fold cross-validation randomly partitions the data set into ten mutually exclusive subsets. We consider each partition (or fold) as the test set and the rest as the training set. We perform feature selection and clustering on the training set, and compute class error on the test set. For each FSSEM variant, the reported error is the average and standard deviation values from the ten-fold cross-validation runs.

Bayes Error: Since we know the true probability distributions for the synthetic data, we provide the Bayes error (Duda et al., 2001) values to give us the lowest average class error rate achievable for these data sets. Instead of a full integration of the error in possibly discontinuous decision regions in multivariate space, we compute the Bayes error experimentally. Using the relevant features and their true distributions, we classify the generated data with an optimal Bayes classifier and calculate the error.

To evaluate the algorithm's ability to select "relevant" features, we report the average number of features selected, and the average feature recall and precision. Recall and precision are concepts from text retrieval (Salton and McGill, 1983) and are defined here as:

Recall: the number of relevant features in the selected subset divided by the total number of relevant features.

Precision: the number of relevant features in the selected subset divided by the total number of features selected.

These measures give us an indication of the quality of the features selected. High values of precision and recall are desired. Feature precision also serves as a measure of how well our dimension normalization scheme (a.k.a. our stopping criterion) works. Finally, to evaluate the clustering algorithm's ability to find the "correct" number of clusters, we report the average number of clusters found.

6.3 Initializing EM and Other Implementation Details

In the EM algorithm, we start with an initial estimate of our parameters, $\Phi^{(0)}$, and then iterate using the update equations until convergence. *Note that EM is initialized for each new feature subset.*

The EM algorithm can get stuck at a local maximum, hence the initialization values are important. We used the sub-sampling initialization algorithm proposed by Fayyad et al. (1998) with 10% sub-sampling and $J = 10$ sub-sampling iterations. Each sub-sample, S_i ($i = 1, \ldots, J$), is randomly initialized. We run k-means (Duda et al., 2001) on these sub-samples not permitting empty clusters (i.e., when an empty cluster exists at the end of k-means, we reset the empty cluster's mean equal to the data furthest from its cluster centroid, and re-run k-means). Each sub-sample results in a set of cluster centroids CM_i, $i, \ldots, J$. We then cluster the combined set, CM, of all CM_i's using k-means initialized by CM_i resulting in new centroids FM_i. We select the FM_i, $i = 1, \ldots, J$, that maximizes the likelihood of CM as our initial clusters.

After initializing the parameters, EM clustering iterates until convergence (i.e., the likelihood does not change by 0.0001) or up to n (default 500) iterations whichever comes first. We limit

the number of iterations because EM converges very slowly near a maximum. We avoid problems with handling singular matrices by adding a scalar ($\delta = 0.000001\sigma^2$, where σ^2 is the average of the variances of the unclustered data) multiplied to the identity matrix (δI) to each of the component covariance matrices Σ_j. This makes the final matrix positive definite (i.e., all eigenvalues are greater than zero) and hence nonsingular. We constrain our solution away from spurious clusters by deleting clusters with any diagonal element equal to or less than δ.

6.4 Experiments on Gaussian Mixture Data

We investigate the biases and compare the performance of the different feature selection criteria. We refer to FSSEM using the separability criterion as FSSEM-TR and using ML as FSSEM-ML. Aside from evaluating the performance of these algorithms, we also report the performance of EM (clustering using all the features) to see whether or not feature selection helped in finding more "interesting" structures (i.e., structures that reveal class labels). FSSEM and EM assume a fixed number of clusters, *k*, equal to the number of classes. We refer to EM clustering and FSSEM with finding the number of clusters as EM-k and FSSEM-k respectively. Due to clarity purposes and space constraints, we only present the relevant tables here. We report the results for all of the evaluation measures presented in Section 6.2 in (Dy and Brodley, 2003).

6.4.1 ML VERSUS TRACE

We compare the performance of the different feature selection criteria (FSSEM-k-TR and FSSEM-k-ML) on our synthetic data. We use FSSEM-k rather than FSSEM, because Section 6.4.3 shows that feature selection with finding *k* (FSSEM-k) is better than feature selection with fixed *k* (FSSEM). Table 1 shows the cross-validated (CV) error and average number of clusters results for *trace* and ML on the five data sets.

Percent CV Error					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-k-TR	4.6 ± 2.0	21.4 ± 06.0	4.2 ± 2.3	3.0 ± 1.8	0.0 ± 0.0
FSSEM-k-ML	55.6 ± 3.9	54.8 ± 17.4	79.4 ± 6.1	84.0 ± 4.1	78.2 ± 6.1
Average Number of Clusters					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-k-TR	2.0 ± 0.0	3.0 ± 0.0	4.0 ± 0.0	5.0 ± 0.0	5.0 ± 0.0
FSSEM-k-ML	1.0 ± 0.0	1.4 ± 0.8	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0

Table 1: Cross-validated error and average number of clusters for FSSEM-k-TR versus FSSEM-k-ML applied to the simulated Gaussian mixture data.

FSSEM-k-TR performed better than FSSEM-k-ML in terms of CV error. Trace performed better than ML, because it selected the features with high cluster separation. ML preferred features with low variance. When the variance of each cluster is the same, ML prefers the feature subset with fewer clusters (which happens to be our noise features). This bias is reflected by an average feature recall of 0.04. FSSEM-k-TR, on the other hand, was biased toward separable clusters identified by our defined relevant features, reflected by an average feature recall of 0.8.

6.4.2 RAW DATA VERSUS STANDARDIZED DATA

In the previous subsection, ML performed worse than *trace* for our synthetic data, because ML prefers features with low variance and fewer clusters (our noise features have lower variance than the relevant features). In this subsection, we investigate whether standardizing the data in each dimension (i.e., normalizing each dimension to yield a variance equal to one) would eliminate this bias. Standardizing data is sometimes done as a pre-processing step in data analysis algorithms to equalize the weight contributed by each feature. We would also like to know how standardization affects the performance of the other FSSEM variants.

Let X be a random data vector and X_f ($f = 1 \ldots d$) be the elements of the vector, where d is the number of features. We standardize X by dividing each element by the corresponding feature standard deviation (X_f/σ_f, where σ_f is the standard deviation for feature f).

Table 2 reports the CV error. Additional experimental results can be found in (Dy and Brodley, 2003). Aside from the FSSEM variants, we examine the effect of standardizing data on EM-k, clustering with finding the number of clusters using all the features. We represent the corresponding variant on standardized data with the suffix "-STD". The results show that only FSSEM-k-ML is affected by standardizing data. The *trace* criterion computes the between-class scatter normalized by the average within-class scatter and is invariant to any linear transformation. Since standardizing data is a linear transformation, the *trace* criterion results remain unchanged.

Standardizing data improves ML's performance. It eliminates ML's bias to lower overall variance features. Assuming equal variance clusters, ML prefers a single Gaussian cluster over two well-separated Gaussian clusters. But, after standardization, the two Gaussian clusters become more favorable because each of the two clusters now has lower variance (i.e., higher probabilities) than the single cluster noise feature. Observe that when we now compare FSSEM-k-TR-STD or FSSEM-k-TR with FSSEM-k-ML-STD, the performance is similar for all our data sets. These results show that scale invariance is an important property for a feature evaluation criterion. If a criterion is not scale invariant such as ML, in this case, pre-processing by standardizing the data in each dimension is necessary. Scale invariance can be incorporated to the ML criterion by modifying the function as presented in (Dy and Brodley, 2003). Throughout the rest of the paper, we standardize the data before feature selection and clustering.

Percent CV Error					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-k-TR	4.6 ± 2.0	21.4 ± 06.0	4.2 ± 2.3	3.0 ± 1.8	0.0 ± 0.0
FSSEM-k-TR-STD	4.6 ± 2.0	21.6 ± 05.4	4.0 ± 2.0	3.0 ± 1.8	0.0 ± 0.0
FSSEM-k-ML	55.6 ± 3.9	54.8 ± 17.4	79.4 ± 6.1	84.0 ± 4.1	78.2 ± 6.1
FSSEM-k-ML-STD	4.8 ± 1.8	21.4 ± 05.1	4.0 ± 2.2	15.2 ± 7.3	0.0 ± 0.0
EM-k	55.6 ± 3.9	63.6 ± 06.0	48.6 ± 9.5	84.0 ± 4.1	55.4 ± 5.5
EM-k-STD	55.6 ± 3.9	63.6 ± 06.0	48.6 ± 9.5	84.0 ± 4.1	56.2 ± 6.1

Table 2: Percent CV error of FSSEM variants on standardized and raw data.

6.4.3 FEATURE SEARCH WITH FIXED k VERSUS SEARCH FOR k

In Section 3, we illustrated that different feature subsets have different numbers of clusters, and that to model the clusters during feature search correctly, we need to incorporate finding the number

of clusters, k, in our approach. In this section, we investigate whether finding k yields better performance than using a fixed number of clusters. We represent the FSSEM and EM variants using a fixed number of clusters (equal to the known classes) as FSSEM and EM. FSSEM-k and EM-k stand for FSSEM and EM with searching for k. Tables 3 and 4 summarize the CV error, average number of cluster, feature precision and recall results of the different algorithms on our five synthetic data sets.

Percent CV Error					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-TR-STD	4.4 ± 02.0	37.6 ± 05.6	7.4 ± 11.0	21.2 ± 20.7	14.4 ± 22.2
FSSEM-k-TR-STD	4.6 ± 02.0	21.6 ± 05.4	4.0 ± 02.0	3.0 ± 01.8	0.0 ± 00.0
FSSEM-ML-STD	7.8 ± 05.5	22.8 ± 06.6	3.6 ± 01.7	15.4 ± 09.5	4.8 ± 07.5
FSSEM-k-ML-STD	4.8 ± 01.8	21.4 ± 05.1	4.0 ± 02.2	15.2 ± 07.3	0.0 ± 00.0
EM-STD	22.4 ± 15.1	30.8 ± 13.1	23.2 ± 10.1	48.2 ± 07.5	10.2 ± 11.0
EM-k-STD	55.6 ± 03.9	63.6 ± 06.0	48.6 ± 09.5	84.0 ± 04.1	56.2 ± 06.1
Bayes	5.4 ± 00.0	20.4 ± 00.0	3.4 ± 00.0	0.8 ± 00.0	0.0 ± 00.0
Average Number of Clusters					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-TR-STD	fixed at 2	fixed at 3	fixed at 4	fixed at 5	fixed at 5
FSSEM-k-TR-STD	2.0 ± 0.0	3.0 ± 0.0	4.0 ± 0.0	5.0 ± 0.0	5.0 ± 0.0
FSSEM-ML-STD	fixed at 2	fixed at 3	fixed at 4	fixed at 5	fixed at 5
FSSEM-k-ML-STD	2.0 ± 0.0	3.0 ± 0.0	4.0 ± 0.0	4.2 ± 0.4	5.0 ± 0.0
EM-STD	fixed at 2	fixed at 3	fixed at 4	fixed at 5	fixed at 5
EM-k-STD	1.0 ± 0.0	1.0 ± 0.0	2.0 ± 0.0	1.0 ± 0.0	2.1 ± 0.3

Table 3: Percent CV error and average number of cluster results on FSSEM and EM with fixed number of clusters versus finding the number of clusters.

Looking first at FSSEM-k-TR-STD compared to FSSEM-TR-STD, we see that including order identification (FSSEM-k-TR-STD) with feature selection results in lower CV error for the *trace* criterion. For all data sets except the two-class data, FSSEM-k-TR-STD had significantly lower CV error than FSSEM-TR-STD. Adding the search for k within the feature subset selection search allows the algorithm to find the relevant features (an average of 0.796 feature recall for FSSEM-k-TR-STD versus 0.656 for FSSEM-TR-STD).[3] This is because the best number of clusters depends on the chosen feature subset. For example, on closer examination, we noted that on the three-class problem when k is fixed at three, the clusters formed by feature 1 are better separated than clusters that are formed by features 1 and 2 together. As a consequence, FSSEM-TR-STD did not select feature 2. When k is made variable during the feature search, FSSEM-k-TR-STD finds two clusters in feature 1. When feature 2 is considered with feature 1, three or more clusters are found resulting in higher separability.

In the same way, FSSEM-k-ML-STD was better than fixing k, FSSEM-ML-STD, for all data sets in terms of CV error except for the four-class data. FSSEM-k-ML-STD performed slightly better than FSSEM-ML-STD for all the data sets in terms of feature precision and recall. This

3. Note that the recall value is low for the five-class fifteen-features data. This is because some of the "relevant" features are redundant as reflected by the 0.0% CV error obtained by our feature selection algorithms.

Average Feature Precision					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-TR-STD	0.62 ± 0.26	0.56 ± 0.24	0.68 ± 0.17	0.95 ± 0.15	1.00 ± 0.00
FSSEM-k-TR-STD	0.57 ± 0.23	0.65 ± 0.05	0.53 ± 0.07	1.00 ± 0.00	1.00 ± 0.00
FSSEM-ML-STD	0.24 ± 0.05	0.52 ± 0.17	0.53 ± 0.10	0.98 ± 0.05	1.00 ± 0.00
FSSEM-k-ML-STD	0.33 ± 0.00	0.67 ± 0.13	0.50 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
EM-k	0.20 ± 0.00	0.20 ± 0.00	0.20 ± 0.00	0.25 ± 0.00	0.75 ± 0.00
EM-k-STD	0.20 ± 0.00	0.20 ± 0.00	0.20 ± 0.00	0.25 ± 0.00	0.75 ± 0.00
Average Feature Recall					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-TR-STD	1.00 ± 0.00	0.55 ± 0.15	0.95 ± 0.15	0.46 ± 0.20	0.32 ± 0.19
FSSEM-k-TR-STD	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.62 ± 0.06	0.36 ± 0.13
FSSEM-ML-STD	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.74 ± 0.13	0.41 ± 0.20
FSSEM-k-ML-STD	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.72 ± 0.16	0.51 ± 0.14
EM-k	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
EM-k-STD	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00

Table 4: Average feature precision and recall obtained by FSSEM with a fixed number of clusters versus FSSEM with finding the number of clusters.

shows that incorporating finding k helps in selecting the "relevant" features. EM-STD had lower CV error than EM-k-STD due to prior knowledge about the correct number of clusters. Both EM-STD and EM-k-STD had poorer performance than FSSEM-k-TR/ML-STD, because of the retained noisy features.

6.4.4 FEATURE CRITERION NORMALIZATION VERSUS WITHOUT NORMALIZATION

Percent CV Error					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-k-TR-STD-notnorm	4.6 ± 2.0	23.4 ± 6.5	4.2 ± 2.3	2.6 ± 1.3	0.0 ± 0.0
FSSEM-k-TR-STD	4.6 ± 2.0	21.6 ± 5.4	4.0 ± 2.0	3.0 ± 1.8	0.0 ± 0.0
FSSEM-k-ML-STD-notnorm	4.6 ± 2.2	36.2 ± 4.2	48.2 ± 9.4	63.6 ± 4.9	46.8 ± 6.2
FSSEM-k-ML-STD	4.8 ± 1.8	21.4 ± 5.1	4.0 ± 2.2	15.2 ± 7.3	0.0 ± 0.0
Bayes	5.4 ± 0.0	20.4 ± 0.0	3.4 ± 0.0	0.8 ± 0.0	0.0 ± 0.0
Average Number of Features Selected					
Method	2-Class	3-Class	4-Class	5-Class, 5-Feat.	5-Class, 15-Feat.
FSSEM-k-TR-STD-notnorm	2.30 ± 0.46	3.00 ± 0.00	3.90 ± 0.30	3.30 ± 0.46	9.70 ± 0.46
FSSEM-k-TR-STD	2.00 ± 0.63	3.10 ± 0.30	3.80 ± 0.40	3.10 ± 0.30	5.40 ± 1.96
FSSEM-k-ML-STD-notnorm	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
FSSEM-k-ML-STD	3.00 ± 0.00	3.10 ± 0.54	4.00 ± 0.00	3.60 ± 0.80	7.70 ± 2.10

Table 5: Percent CV error and average number of features selected by FSSEM with criterion normalization versus without.

Table 5 presents the CV error and average number of features selected by feature selection with cross-projection criterion normalization versus without (those with suffix "notnorm"). Here and throughout the paper, we refer to normalization as the feature normalization scheme (cross-projection method) described in Section 5. For the *trace* criterion, without normalization did not affect the CV error. However, normalization achieved similar CV error performance using fewer features than without normalization. For the ML criterion, criterion normalization is definitely needed. Note that without, FSSEM-k-ML-STD-notnorm selected only a single feature for each data set resulting in worse CV error performance than with normalization (except for the two-class data which has only one relevant feature).

6.4.5 FEATURE SELECTION VERSUS WITHOUT FEATURE SELECTION

In all cases, feature selection (FSSEM, FSSEM-k) obtained better results than without feature selection (EM, EM-k) as reported in Table 3. Note that for our data sets, the noise features misled EM-k-STD, leading to fewer clusters than the "true" k. Observe too that FSSEM-k was able to find approximately the true number of clusters for the different data sets.

In this subsection, we experiment on the sensitivity of the FSSEM variants to the number of noise features. Figures 9a-e plot the cross-validation error, average number of clusters, average number of noise features, feature precision and recall respectively of feature selection (FSSEM-k-TR-STD and FSSEM-k-ML-STD) and without feature selection (EM-k-STD) as more and more noise features are added to the four-class data. Note that the CV error performance, average number of clusters, average number of selected features and feature recall for the feature selection algorithms are more or less constant throughout and are approximately equal to clustering with no noise. The feature precision and recall plots reveal that the CV error performance of feature selection was not affected by noise, because the FSSEM-k variants were able to select the relevant features (recall $= 1$) and discard the noisy features (high precision). Figure 9 demonstrates the need for feature selection as irrelevant features can mislead clustering results (reflected by EM-k-STD's performance as more and more noise features are added).

6.4.6 CONCLUSIONS ON EXPERIMENTS WITH SYNTHETIC DATA

Experiments on simulated Gaussian mixture data reveal that:

- Standardizing the data before feature subset selection in conjunction with the ML criterion is needed to remove ML's preference for low variance features.
- Order identification led to better results than fixing k, because different feature subsets have different number of clusters as illustrated in Section 3.
- The criterion normalization scheme (cross-projection) introduced in Section 5 removed the biases of *trace* and ML with respect to dimension. The normalization scheme enabled feature selection with *trace* to remove "redundant" features and prevented feature selection with ML from selecting only a single feature (a trivial result).
- Both ML and *trace* with feature selection performed equally well for our five data sets. Both criteria were able to find the "relevant" features.
- Feature selection obtained better results than without feature selection.

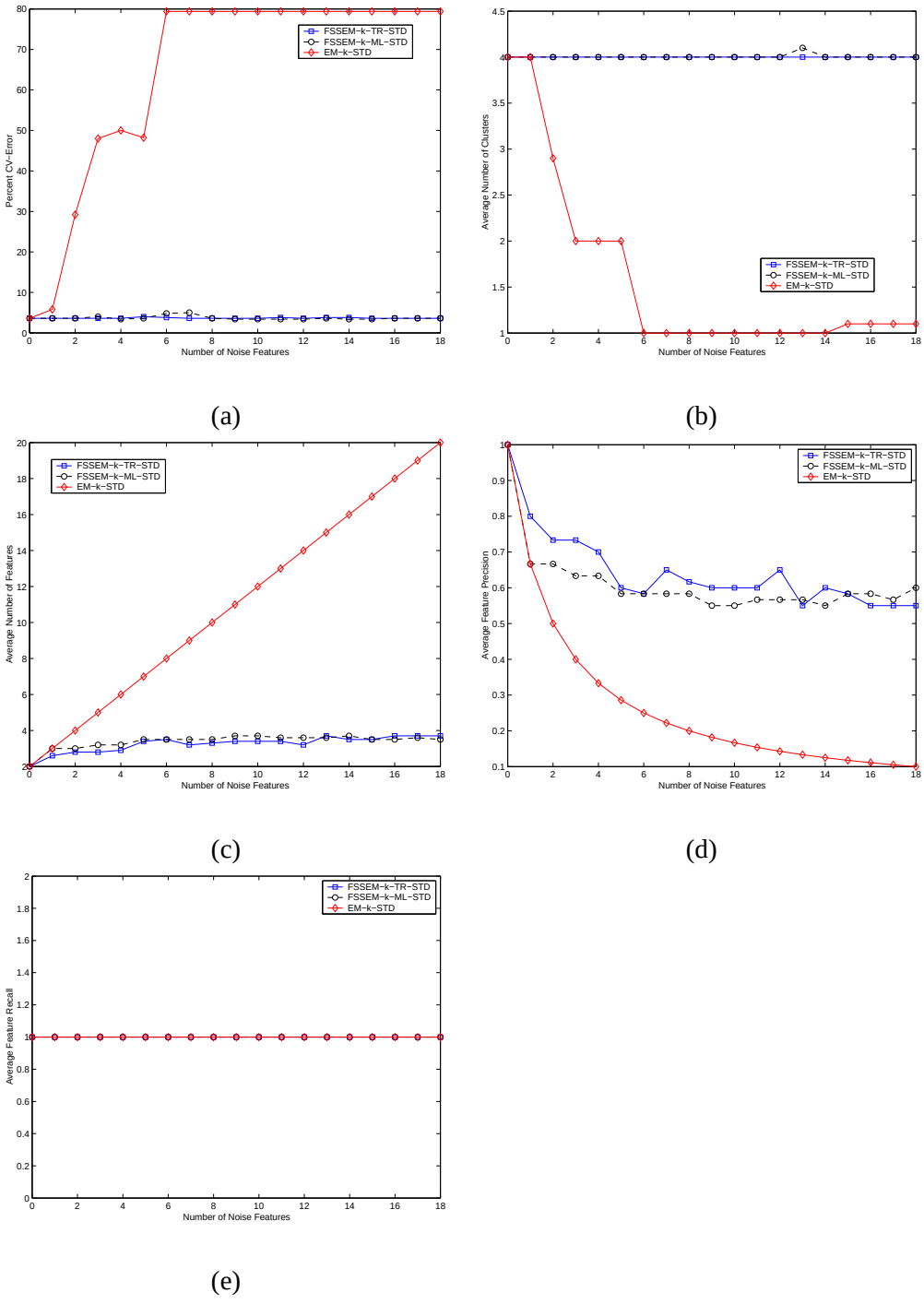


Figure 9: Feature selection versus without feature selection on the four-class data.

6.5 Experiments on Real Data

We examine the FSSEM variants on the iris, wine, and ionosphere data set from the UCI learning repository (Blake and Merz, 1998), and on a high resolution computed tomography (HRCT) lung

image data which we collected from IUPUI medical center (Dy et al., 2003; Dy et al., 1999). Although for each data set the class information is known, we remove the class labels during training.

Unlike synthetic data, we do not know the "true" number of (Gaussian) clusters for real-world data sets. Each class may be composed of many Gaussian clusters. Moreover, the clusters may not even have a Gaussian distribution. To see whether the clustering algorithms found clusters that correspond to classes (wherein a class can be multi-modal), we compute the cross-validated class error in the same way as for the synthetic Gaussian data. On real data sets, we do not know the "relevant" features. Hence, we cannot compute precision and recall and therefore report only the average number of features selected and the average number of clusters found.

Although we use class error as a measure of cluster performance, we should not let it misguide us in its interpretation. Cluster quality or interestingness is difficult to measure because it depends on the particular application. This is a major distinction between unsupervised clustering and supervised learning. Here, class error is just *one interpretation of the data.* We can also measure cluster performance in terms of the *trace* criterion and the ML criterion. Naturally, FSSEM-k-TR and FSSEM-TR performed best in terms of *trace*; and, FSSEM-k-ML and FSSEM-ML were best in terms of maximum likelihood. Choosing either TR or ML depends on your application goals. If you are interested in finding the features that best separate the data, use FSSEM-k-TR. If you are interested in finding features that model Gaussian clusters best, use FSSEM-k-ML.

To illustrate the generality and ease of applying other clustering methods in the wrapper framework, we also show the results for different variants of feature selection wrapped around the *k*-means clustering algorithm (Forgy, 1965; Duda et al., 2001) coupled with the TR and ML criteria. We use sequential forward search for feature search. To find the number of clusters, we apply the BIC penalty criterion (Pelleg and Moore, 2000). We use the following acronyms throughout the rest of the paper: Kmeans stands for the *k*-means algorithm, FSS-Kmeans stands for feature selection wrapped around *k*-means, TR represents the *trace* criterion for feature evaluation, ML represents ML criterion for evaluating features, "-k-" represents that the variant finds the number of clusters, and "-STD" shows that the data was standardized such that each feature has variance equal to one.

Since cluster quality depends on the initialization method used for clustering, we performed EM clustering using three different initialization methods:

1. Initialize using ten *k*-means starts with each *k*-means initialized by a random seed, then pick the final clustering corresponding to the highest likelihood.

2. Ten random re-starts.

3. Fayyad et al.'s method as described earlier in Section 6.3 (Fayyad et al., 1998).

Items one and two are similar for the *k*-means clustering. Hence, for *k*-means, we initialize with items two and three (with item three performed using Fayyad et al.'s method for *k*-means (Bradley and Fayyad, 1998) which applies *k*-means to the sub-sampled data instead of EM and distortion to pick the best clustering instead of the ML criterion). In the discussion section as follows, we show the results for FSSEM and FSS-Kmeans variants using the initialization which provides consistently good CV-error across all methods. We present the results using each initialization method on all the FSSEM and FSS-Kmeans variants in (Dy and Brodley, 2003) Appendix E. On the tables, "-1", "-2", and "-3" represent the initialization methods 1, 2, and 3 respectively.

Iris Data and FSSEM Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSSEM-TR-STD-1	2.7 ± 04.4	fixed at 3	3.5 ± 0.7
FSSEM-k-TR-STD-1	4.7 ± 05.2	3.1 ± 0.3	2.7 ± 0.5
FSSEM-ML-STD-1	7.3 ± 12.1	fixed at 3	3.6 ± 0.9
FSSEM-k-ML-STD-1	3.3 ± 04.5	3.0 ± 0.0	2.5 ± 0.5
EM-STD-1	3.3 ± 05.4	fixed at 3	fixed at 4
EM-k-STD-1	42.0 ± 14.3	2.2 ± 0.6	fixed at 4
Iris Data and FSS-Kmeans Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSS-Kmeans-TR-STD-2	2.7 ± 03.3	fixed at 3	1.9 ± 0.3
FSS-Kmeans-k-TR-STD-2	13.3 ± 09.4	4.5 ± 0.7	2.3 ± 0.5
FSS-Kmeans-ML-STD-2	2.0 ± 03.1	fixed at 3	2.0 ± 0.0
FSS-Kmeans-k-ML-STD-2	4.7 ± 04.3	3.4 ± 0.5	2.4 ± 0.5
Kmeans-STD-2	17.3 ± 10.8	fixed at 3	fixed at 4
Kmeans-k-STD-2	44.0 ± 11.2	2.0 ± 0.0	fixed at 4

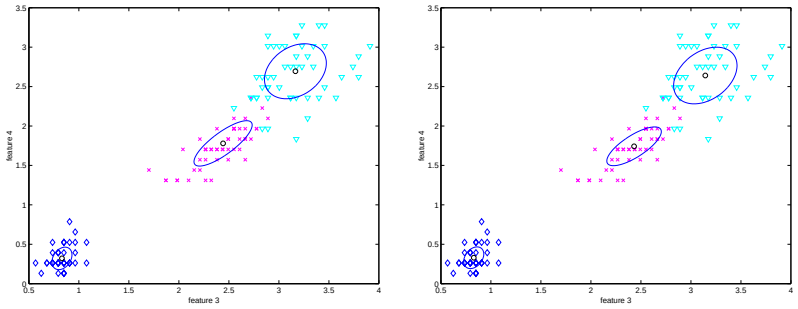
Table 6: Results for the different variants on the iris data.

6.5.1 IRIS DATA

We first look at the simplest case, the Iris data. This data has three classes, four features, and 150 instances. Fayyad et. al's method of initialization works best for large data sets. Since the Iris data only has a few number of instances and classes that are well-separated, ten *k*-means starts provided the consistently best result for initializing EM clustering across the different methods. Table 6 summarizes the results for the different variants of FSSEM compared to EM clustering without feature selection. For the iris data, we set K_{max} in FSSEM-k equal to six, and for FSSEM we fixed k at three (equal to the number of labeled classes). The CV error for FSSEM-k-TR-STD and FSSEM-k-ML-STD are much better than EM-k-STD. This means that when you do not know the "true" number of clusters, feature selection helps find good clusters. FSSEM-k even found the "correct" number of clusters. EM clustering with the "true" number of clusters (EM-STD) gave good results. Feature selection, in this case, did not improve the CV-error of EM-STD, however, they produced similar error rates with fewer features. FSSEM with the different variants consistently chose feature 3 (petal-length), and feature 4 (petal-width). In fact, we learned from this experiment that only these two features are needed to correctly cluster the iris data to three groups corresponding to iris-setosa, iris-versicolor and iris-viginica. Figures 10 (a) and (b) show the clustering results as a scatterplot on the first two features chosen by FSSEM-k-TR and FSSEM-k-ML respectively. The results for feature selection wrapped around *k*-means are also shown in Table 6. We can infer similar conclusions from the results on FSS-Kmeans variants as with the FSSEM variants for this data set.

6.5.2 WINE DATA

The wine data has three classes, thirteen features and 178 instances. For this data, we set K_{max} in FSSEM-k equal to six, and for FSSEM we fixed k at three (equal to the number of labeled classes). Table 7 summarizes the results when FSSEM and the FSS-Kmeans variants are initialized with ten *k*-means starts and ten random re-starts respectively. These are the initialization methods which led to the best performance for EM and *k*-means without feature selection. When "k" is



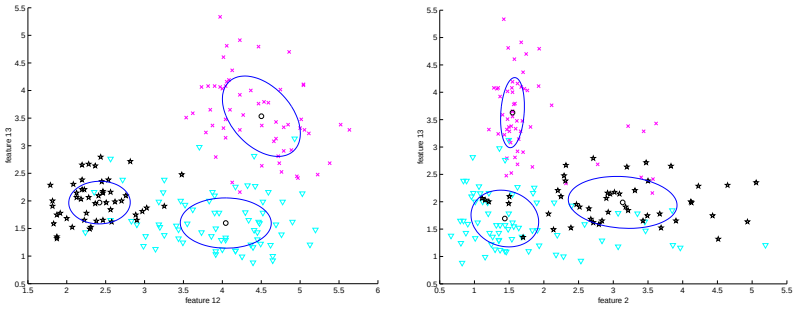
(a) (b)

Figure 10: The scatter plots on iris data using the first two features chosen by FSSEM-k-TR (a) and FSSEM-k-ML (b). $\diamond$, $\times$ and $\bigtriangledown$ represent the different class assignments. $\circ$ are the cluster means, and the ellipses are the covariances corresponding to the clusters discovered by FSSEM-k-TR and FSSEM-k-ML.

Wine Data and FSSEM Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSSEM-TR-STD-1	44.0 ± 08.1	fixed at 3	1.4 ± 0.5
FSSEM-k-TR-STD-1	12.4 ± 13.0	3.6 ± 0.8	3.8 ± 1.8
FSSEM-ML-STD-1	30.6 ± 21.8	fixed at 3	2.9 ± 0.8
FSSEM-k-ML-STD-1	23.6 ± 14.4	3.9 ± 0.8	3.0 ± 0.8
EM-STD-1	10.0 ± 17.3	fixed at 3	fixed at 13
EM-k-STD-1	37.1 ± 12.6	3.2 ± 0.4	fixed at 13
Wine Data and FSS-Kmeans Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSS-Kmeans-TR-STD-2	37.3 ± 14.0	fixed at 3	1.0 ± 0.0
FSS-Kmeans-k-TR-STD-2	28.1 ± 09.6	3.6 ± 0.5	2.5 ± 0.9
FSS-Kmeans-ML-STD-2	16.1 ± 09.9	fixed at 3	3.1 ± 0.3
FSS-Kmeans-k-ML-STD-2	18.5 ± 07.2	4.2 ± 0.6	3.1 ± 0.7
Kmeans-STD-2	0.0 ± 00.0	fixed at 3	fixed at 13
Kmeans-k-STD-2	33.4 ± 21.3	2.6 ± 0.8	fixed at 13

Table 7: Results for the different variants on the wine data set.

known, *k*-means was able to find the clusters corresponding to the "true" classes correctly. EM clustering also performed well when "k" is given. EM and *k*-means clustering performed poorly in terms of CV error when "k" is unknown. It is in this situation where feature selection, FSSEM-k and FSS-Kmeans-k, helped the base clustering methods find good groupings. Interestingly, for the wine data, FSSEM-k-TR performed better than FSSEM-k-ML, and FSS-Kmeans-ML had better CV-error than FSS-Kmeans-TR. This is an example on where using different criteria for feature selection and clustering improved the results through their interaction. Figures 11 (a) and (b) show the scatterplots and clusters discovered projected on the first two features chosen by FSSEM-k-TR and FSS-Kmeans-k-ML respectively. FSSEM-k-TR picked features $\{12, 13, 7, 5, 10, 1, 4\}$ and



(a) (b)

Figure 11: The scatter plots on the wine data using the first two features chosen by FSSEM-k-TR (a) and FSSEM-k-ML (b). $\star$, $\times$ and $\bigtriangledown$ represent the different class assignments. $\circ$ are the cluster means, and the ellipses are the covariances corresponding to the clusters discovered by FSSEM-k-TR and FSS-Kmeans-k-ML.

FSS-Kmeans-k-ML selected features $\{2, 13, 12\}$.[4] Features 12 and 13 stand for "OD280-OD315 of diluted wines" and "proline."

6.5.3 IONOSPHERE DATA

The radar data is collected from a phased array of sixteen high-frequency antennas. The targets are free electrons in the atmosphere. Classes label the data as either good (radar returns showing structure in the ionosphere) or bad returns. There are 351 instances with 34 continuous attributes (measuring time of pulse and pulse number). Features 1 and 2 are discarded, because their values are constant or discrete for all instances. Constant feature values produce an infinite likelihood value for a Gaussian mixture model. Discrete feature values with discrete levels less than or equal to the number of clusters also produce an infinite likelihood value for a finite Gaussian mixture model.

Table 8 reports the ten-fold cross-validation error and the number of clusters found by the different EM and FSSEM algorithms. For the ionosphere data, we set K_{max} in FSSEM-k equal to ten, and fixed k at two (equal to the number of labeled classes) in FSSEM. FSSEM-k-ML and EM clustering with "k" known performed better in terms of CV error compared to the rest of the EM variants. Note that FSSEM-k-ML gave comparable performance with EM using fewer features and with no knowledge of the "true" number of clusters. Table 8 also shows the results for the different k-means variants. FSS-Kmeans-k-ML-STD obtains the best CV error followed closely by FSS-Kmeans-ML-STD. Interestingly, these two methods and FSSEM-k-ML all chose features 5 and 3 (based on the original 34 features) as their first two features.

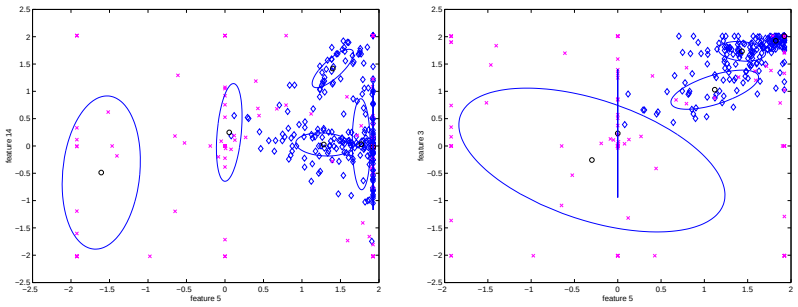
Figures 12a and b present scatterplots of the ionosphere data on the first two features chosen by FSSEM-k-TR and FSSEM-k-ML together with their corresponding means (in $\circ$'s) and covariances (in ellipses) discovered. Observe that FSSEM-k-TR favored the clusters and features in Figure 12a because the clusters are well separated. On the other hand, FSSEM-k-ML favored the clusters in Figure 12b, which have small generalized variances. Since the ML criterion matches the ionosphere

4. These feature subsets are the features which provided the best CV-error performance among the ten-fold runs.

Ionosphere Data and FSSEM Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSSEM-TR-STD-2	38.5 ± 06.5	fixed at 2	2.3 ± 0.9
FSSEM-k-TR-STD-2	23.1 ± 05.8	6.6 ± 1.3	1.1 ± 0.3
FSSEM-ML-STD-2	37.9 ± 07.5	fixed at 2	2.7 ± 2.1
FSSEM-k-ML-STD-2	18.8 ± 06.9	7.6 ± 1.0	2.9 ± 1.1
EM-STD-2	16.8 ± 07.3	fixed at 2	fixed at 32
EM-k-STD-2	35.3 ± 10.3	8.4 ± 1.0	fixed at 32
Ionosphere Data and FSS-Kmeans Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSS-Kmeans-TR-STD-2	35.3 ± 06.5	fixed at 2	1.0 ± 0.0
FSS-Kmeans-k-TR-STD-2	22.8 ± 08.5	9.8 ± 0.4	1.0 ± 0.0
FSS-Kmeans-ML-STD-2	17.7 ± 04.9	fixed at 2	3.5 ± 0.8
FSS-Kmeans-k-ML-STD-2	16.2 ± 04.8	9.3 ± 0.8	1.7 ± 0.8
Kmeans-STD-2	23.4 ± 10.1	fixed at 2	fixed at 32
Kmeans-k-STD-2	28.8 ± 10.8	7.7 ± 0.6	fixed at 32

Table 8: Results for the different variants on the ionosphere data set.

class labels more closely, FSSEM-k-ML performed better with respect to CV error. FSSEM-k-ML obtained better CV error than EM-k; FSS-Kmeans-ML and FSS-Kmeans-k-ML also performed better than Kmeans and Kmeans-k in terms of CV error. The feature selection variants performed better using fewer features compared to the 32 features used by EM-k, Kmeans, and Kmeans-k. It is interesting to note that for this data, random re-start initialization obtained significantly better CV error for EM clustering (16.8%) compared to the other initialization methods (20.5% and 24.8% for ten *k*-means starts and Fayyad et al.'s method respectively). This is because the two "true" classes are highly overlapped. Ten *k*-means starts tend to start-off with well-separated clusters.



(a) (b)

Figure 12: The scatter plots on the ionosphere data using the first two features chosen by FSSEM-k-TR (a) and FSSEM-k-ML (b). $\times$ and $\diamond$ represent the different class assignments. $\circ$ are the cluster means, and the ellipses are the covariances corresponding to the clusters discovered by FSSEM-k-TR and FSSEM-k-ML.

6.5.4 HRCT-LUNG DATA

(a) Centrilobular Emphysema (b) Paraseptal Emphysema (c) IPF

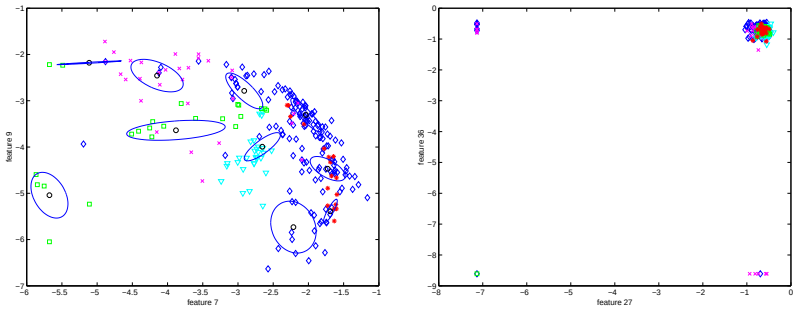
Figure 13: HRCT-lung images.

HRCT-lung consists of 500 instances. Each of these instances are represented by 110 low-level continuous features measuring geometric, gray level and texture features (Dy et al., 1999). We actually used only 108 features because two of the features are constant or discrete. We also log-transformed the data to make our features which are mostly positive real-valued numbers more Gaussian. For features with negative values (like the feature, local mean minus global mean), we add an offset making the minimum value equal to zero. We assign $\log(0)$ to be $\log(0.0000000000001)$. The data is classified into five disease classes (Centrilobular Emphysema, Paraseptal Emphysema, EG, IPF, and Panacinar). Figure 13 shows three HRCT-lung images from three of the disease classes. The white marking is the pathology bearing region (PBR) marked by a radiologist. An instance represents a PBR. An image may contain more than one PBR and more than one disease classification. Note that Centrilobular Emphysema (CE) is characterized by a large number of low intensity (darker) regions which may occupy the entire lung as in Figure 13a. Paraseptal Emphysema (PE) is also characterized by low intensity regions (see Figure 13b). Unlike CE, these regions occur near the boundaries or near fissures. The dark regions are usually separated by thin walls from their adjacent boundary or fissure. CE and PE can be further grouped according to disease severity characterized by the intensity of the regions. Lower intensities indicate more severe cases. The lung image of IPF is characterized by high intensities forming a "glass-like" structure as shown in Figure 13c. Feature selection is important for this data set, because EM clustering using all the features results in just one cluster.

Table 9 presents the results on the HRCT-lung data set. For the HRCT lung data, FSSEM-k-TR and FSS-Kmeans-k-TR performed better than FSSEM-k-ML and FSS-Kmeans-k-ML respectively in terms of CV error. Figures 14 (a) and (b) present scatterplots of the HRCT-lung data on the first two features chosen by FSSEM-k-TR and FSSEM-k-ML. Observe that the clusters found by FSSEM-k-TR are well separated and match the class labels well. FSSEM-k-ML, on the other hand, selects features that result in high-density clusters. Figure 14 (b) demonstrates this clearly. Note also that the "true" number of clusters for this data is more than five (the number of labeled classes). This helped FSSEM-k-TR and FSS-Kmeans-k-TR obtained better results than their fixed-k variants.

HRCT-Lung Data and FSSEM Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSSEM-TR-STD-3	36.8 ± 6.6	fixed at 5	1.3 ± 0.5
FSSEM-k-TR-STD-3	26.6 ± 7.7	6.0 ± 2.7	1.7 ± 0.9
FSSEM-ML-STD-3	37.2 ± 5.5	fixed at 5	3.3 ± 0.6
FSSEM-k-ML-STD-3	37.0 ± 5.7	5.2 ± 1.7	6.6 ± 2.8
EM-STD-3	37.2 ± 5.5	fixed at 5	fixed at 108
EM-k-STD-3	37.2 ± 5.5	1.1 ± 0.3	fixed at 108
HRCT-Lung Data and FSS-Kmeans Variants			
Method	%CV Error	Ave. No. of Clusters	Ave. No. of Features
FSS-Kmeans-TR-STD-3	37.2 ± 05.5	fixed at 5	1.0 ± 0.0
FSS-Kmeans-k-TR-STD-3	28.0 ± 10.7	7.5 ± 1.9	2.9 ± 2.3
FSS-Kmeans-ML-STD-3	36.8 ± 05.9	fixed at 5	3.4 ± 0.7
FSS-Kmeans-k-ML-STD-3	35.6 ± 06.7	4.3 ± 0.9	5.8 ± 3.1
Kmeans-STD-3	36.6 ± 04.9	fixed at 5	fixed at 108
Kmeans-k-STD-3	37.0 ± 05.3	3.4 ± 0.5	fixed at 108

Table 9: Results on the HRCT-lung image data set for the different variants.



(a) (b)

Figure 14: The scatter plots on the HRCT-lung data using the first two features chosen by FSSEM-k-TR (a) and FSSEM-k-ML (b). $\times$, $\diamond$, $\square$, $*$, and $\bigtriangledown$ represent the different class assignments. $\circ$ are the cluster means, and the ellipses are the covariances corresponding to the clusters discovered by FSSEM-k-TR.

HRCT-lung is a difficult data set due to its skewed class distribution (approximately 62.8% of the data is from the disease Centrilobular Emphysema). Because of this, even though EM-k discovered approximately only one cluster, its class error (which is equal to the error using a majority classification rule) is close to the values obtained by the other methods. The high dimensions obscure the HRCT-lung's classes and result in EM-k finding only one cluster. Even with a difficult problem such as this, feature selection obtained better CV-error than without feature selection using much fewer features (an average of 1.7 for FSSEM-k-TR and 2.9 for FSS-Kmeans-k-TR) compared to the original 108 features. FSSEM-k-TR picked features $\{7, 9\}$ and FSS-Kmeans-k-TR chose features $\{8, 6, 59\}$. Features $6, 7, 8$, and 9 are gray level histogram values of the lung region, and feature 59 is a histogram value at a local pathology bearing region. These features make sense in discriminating

between Centrilobular Emphysema (the largest class) from the rest, as this class is characterized by low gray level values.

6.5.5 CONCLUSIONS ON EXPERIMENTS WITH REAL DATA

Our results on real data show that feature selection improved the performance of clustering algorithms in finding "interesting" patterns. We measure "interestingness" performance here by how well the discovered clusters match labeled classes (CV-error). FSSEM-k and FSS-Kmeans-k obtained better CV-error than EM-k and *k*-means using fewer features. Moreover, our experiments reveal that no one feature selection criterion (ML or TR) is better than the other. They have different biases. ML selects features that results in high-density clusters, and performed better than TR on the ionosphere data. Scatter separability (TR) prefers features that reveal well-separated clusters, and performed better than ML on the HRCT-lung data. They both did well on the iris and wine data.

7. Related Work: A Review of Feature Selection Algorithms for Unsupervised Learning

There are three different ways to select features from unsupervised data: 1) after clustering, 2) before clustering, and 3) during clustering. An example algorithm that performs feature selection after clustering is (Mirkin, 1999). The method first applies a new separate-and-conquer version of *k*-means clustering. Then, it computes the contribution weight of each variable in proportion to the squared deviation of each variable's within-cluster mean from the total mean. It represents clusters by conjunctive concepts starting from the variable with the highest weight, until adding variables (with its conceptual description) does not improve the cluster "precision error". Feature selection after clustering is important for conceptual learning, for describing and summarizing structure from data. This type of selecting features can remove redundancy but not feature irrelevance because the initial clustering is performed using all the features. As pointed out earlier, the existence of irrelevant features can misguide clustering results. Using all the features for clustering also assumes that our clustering algorithm does not break down with high dimensional data. In this paper, we only examine feature selection algorithms that affect (can change) the clustering outcomes; i.e., before or during clustering.

A significant body of research exists on methods for feature subset selection for supervised data. These methods can be grouped as *filter* (Marill and Green, 1963; Narendra and Fukunaga, 1977; Almuallim and Dietterich, 1991; Kira and Rendell, 1992; Kononenko, 1994; Liu and Setiono, 1996; Cardie, 1993; Singh and Provan, 1995) or *wrapper* (John et al., 1994; Doak, 1992; Caruana and Freitag, 1994; Aha and Bankert, 1994; Langley and Sage, 1994; Pazzani, 1995) approaches. To maintain the filter/wrapper model distinction used in supervised learning, we define *filter* methods in unsupervised learning as using some intrinsic property of the data to select features without utilizing the clustering algorithm that will ultimately be applied. *Wrapper* approaches, on the other hand, apply the unsupervised learning algorithm to each candidate feature subset and then evaluate the feature subset by criterion functions that utilize the clustering result.

When we first started this research, not much work has been done in feature subset selection for unsupervised learning in the context of machine learning, although research in the form of principal components analysis (PCA) (Chang, 1983), factor analysis (Johnson and Wichern, 1998) and projection pursuit (Friedman, 1987; Huber, 1985) existed. These early works in data reduction for unsupervised data can be thought of as filter methods, because they select the features prior to apply-

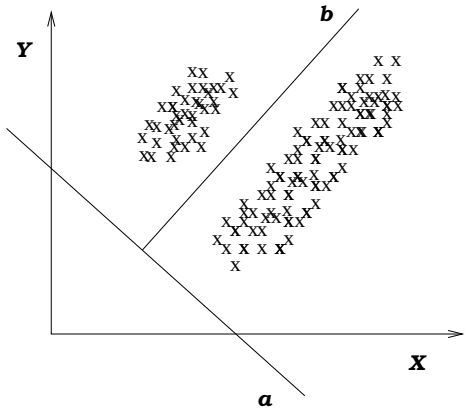


Figure 15: Illustration on when PCA is a poor discriminator.

ing clustering. But rather than selecting a subset of the features, they involve some type of feature transformation. PCA and factor analysis aim to reduce the dimension such that the representation is as faithful as possible to the original data. Note that data reduction techniques based on representation (like PCA) are better suited for compression applications rather than classification (Fukunaga (1990) provides an illustrative example on this). Figure 15 recreates this example. PCA chooses the projection with the highest variance. Projecting two dimensions to one dimension in this example, PCA would project the data to axis *b*, which is clearly inferior to axis *a* for discriminating the two clusters. Contrary to PCA and factor analysis, projection pursuit aims to find "interesting" projections from multi-dimensional data for visualizing structure in the data. A recent method for finding transformations called independent components analysis (ICA) (Hyvärinen, 1999) has gained widespread attention in signal processing. ICA tries to find a transformation such that the transformed variables are statistically independent.

The filter methods described in the previous paragraph all involve transformations of the original variable space. In this paper, we are interested in subsets of the original space, because some domains prefer the original variables in order to maintain the physical interpretation of these features. Moreover, transformations of the variable space require computation or collection of all the features before dimension reduction can be achieved, whereas subsets of the original space require computation or collection of only the selected feature subsets after feature selection is determined. If some features cost more than others, one can consider these costs in selecting features. In this paper, we assume each feature has equal cost. Other interesting and current directions in feature selection involving feature transformations are mixtures of principal component analyzers (Kambhatla and Leen, 1997; Tipping and Bishop, 1999) and mixtures of factor analyzers (Ghahramani and Beal, 2000; Ghahramani and Hinton, 1996; Ueda et al., 1999). We consider these mixture algorithms as wrapper approaches.

In recent years, more attention has been paid to unsupervised feature subset selection. Most of these methods are wrapper approaches. Gennari (1991) incorporates feature selection (they call "attention") to CLASSIT (an incremental concept formation hierarchical clustering algorithm introduced in (Gennari et al., 1989)). The attention algorithm inspects the features starting with the most salient ("per-attribute contribution to category utility") attribute to the least salient attribute,

and stop inspecting features if the remaining features do not change the current clustering decision. The purpose of this attention mechanism is to increase efficiency without loss of prediction accuracy. Devaney and Ram (1997) applied sequential forward and backward search. To evaluate each candidate subset, they measured the category utility of the clusters found by applying COBWEB (Fisher, 1987) in conjunction with the feature subset. Talavera (1999) applied "blind" (similar to the filter) and "feedback" (analogous to the wrapper) approaches to COBWEB, and used a feature dependence measure to select features. Vaithyanathan and Dom (1999) formulated an objective function for choosing the feature subset and finding the optimal number of clusters for a document clustering problem using a Bayesian statistical estimation framework. They modeled each cluster as a multinomial. They extended this concept to create hierarchical clusters (Vaithyanathan and Dom, 2000). Agrawal, et al. (1998) introduced a clustering algorithm (CLIQUE) which proceeds level-by-level from one feature to the highest dimension or until no more feature subspaces with clusters (regions with high density points) are generated. CLIQUE is a density based clustering algorithm which does not assume any density model. However, CLIQUE needs to specify parameters τ (the density threshold) and ν (the equal length interval partitioning for each dimension). In contrast, our method makes assumptions about distributions to avoid specifying parameters. Kim, Street and Menczer (2002) apply an evolutionary local selection algorithm (ELSA) to search the feature subset and number of clusters on two clustering algorithms: K-means and EM clustering (with diagonal covariances), and a Pareto front to combine multiple objective evaluation functions. Law, Figueiredo and Jain (2002) estimate feature saliency using EM by modeling relevant features as conditionally independent given the component label, and irrelevant features with a probability density identical for all components. They also developed a wrapper approach that selects features using Kullback-Leibler divergence and entropy. Friedman and Meulman (2003) designed a distance measure for attribute-value data for clustering on subsets of attributes, and allow feature subsets for each cluster to be different.

8. Summary

In this paper, we introduced a wrapper framework for performing feature subset selection for unsupervised learning. We explored the issues involved in developing algorithms under this framework. We identified the need for finding the number of clusters in feature search and provided proofs for the biases of ML and scatter separability with respect to dimension. We, then, presented methods to ameliorate these problems.

Our experimental results showed that incorporating finding the number of clusters k into the feature subset selection process led to better results than fixing k to be the true number of classes. There are two reasons: 1) the number of classes is not necessarily equal to the number of Gaussian clusters, and 2) different feature subsets have different number of clusters. Supporting theory, our experiments on simulated data showed that ML and scatter separability are in some ways biased with respect to dimension. Thus, a normalization scheme is needed for the chosen feature selection criterion. Our proposed cross-projection criterion normalization scheme was able to eliminate these biases.

Although we examined the wrapper framework using FSSEM, the search method, feature selection criteria (especially the *trace* criterion), and the feature normalization scheme can be easily applied to any clustering method. The issues we have encountered and solutions presented are applicable to any feature subset wrapper approach. FSSEM serves as an example. Depending on one's

application, one may choose to apply a more appropriate search method, clustering and feature selection criteria.

9. Future Directions

Research in feature subset selection for unsupervised learning is quite young. Even though we have addressed some issues, the paper opens up more questions that need to be answered.

Hartigan (1985) pointed out that no single criterion is best for all applications. This is reiterated by our results on the HRCT and Ionosphere data. This led us to work in visualization and user interaction to guide the feature search (Dy and Brodley, 2000b). Another interesting direction is to look at feature selection with hierarchical clustering (Gennari, 1991; Fisher, 1996; Devaney and Ram, 1997; Talavera, 1999; Vaithyanathan and Dom, 2000), since hierarchical clustering provides groupings at various perceptual levels. In addition, a cluster may be modeled better by a different feature subset from other clusters. One may wish to develop algorithms that select a different feature subset for each cluster component.

We explored unsupervised feature selection through the wrapper framework. It would be interesting to do a rigorous investigation of filter versus wrapper approach for unsupervised learning. One may also wish to venture in transformations of the original variable space. In particular, investigate on mixtures of principal component analyzers (Kambhatla and Leen, 1997; Tipping and Bishop, 1999), mixtures of factor analyzers (Ghahramani and Beal, 2000; Ghahramani and Hinton, 1996; Ueda et al., 1999) and mixtures of independent component analyzers (Hyvärinen, 1999).

The difficulty with unsupervised learning is the absence of labeled examples to guide the search. Breiman (Breiman, 2002) suggests transforming the clustering problem into a classification problem by assigning the unlabeled data to class one, and adding the same amount of random vectors into another class two. The second set is generated by independent sampling from the one-dimensional marginal distributions of class one. Understanding and developing tricks such as these to uncover structure from unlabled data remains as topics that need further investigation. Another avenue for future work is to explore semi-supervised (few labeled examples and large amounts of unlabeled data) methods for feature selection.

Finally, in feature selection for unsupervised learning, several fundamental questions are still unanswered:

1. How do you define what "interestingness" means?

2. Should the criterion for "interestingness" (feature selection criterion) be the same as the criterion for "natural" grouping (clustering criterion)? Most of the literature uses the same criterion for feature selection and clustering as this leads to a clean optimization formulation. However, defining "interestingness" into a mathematical criterion is a difficult problem. Allowing different criteria to interact may provide a better model. Our experimental results on the wine data suggest this direction.

3. Our experiments on synthetic data indicate the need to standardize features. Mirkin, 1999, also standardized his features. Should features always be standardized before feature selection? If so, how do you standardize data containing different feature types (real-valued, nominal, and discrete)?

4. What is the best way to evaluate the results? In this paper, we evaluate performance using an external criterion (cross-validated class error). This is a standard measure used by most papers in the feature selection for unsupervised learning literature. Class error is task specific and measures the performance for one labeling solution. Is this the best way to compare different clustering algorithms?

Acknowledgments

The authors wish to thank Dr. Craig Codrington for discussions on criterion normalization, the ML-lunch group at Purdue University for helpful comments, and the reviewers for their constructive remarks. This research is supported by NSF Grant No. IRI9711535, and NIH Grant No. 1 R01 LM06543-01A1.

Appendix A. EM Clustering

Clustering using finite mixture models is a well-known method and has been used for a long time in pattern recognition Duda and Hart (1973); Fukunaga (1990); Jain and Dubes (1988) and statistics McLachlan and Basford (1988); Titterington et al. (1985); Fraley and Raftery (2000). In this model, one assumes that the data is generated from a mixture of component density functions, in which each component density function represents a cluster. The probability distribution function of the data has the following form:

$$f(X_i|\Phi) = \sum_{j=1}^{k} \pi_j f_j(X_i|\theta_j) \tag{5}$$

where $f_j(X_i|\theta_j)$ is the probability density function for class j, π_j is the mixing proportion of class j (prior probability of class j), k is the number of clusters, X_i is a d-dimensional random data vector, θ_j is the set of parameters for class j, $\Phi = (\pi, \theta)$ is the set of all parameters and $f(X_i|\Phi)$ is the probability density function of our observed data point X_i given the parameters Φ. Since the π_j's are prior probabilities, they are subject to the following constraints: $\pi_j \geq 0$ and $\sum_{j=1}^{k} \pi_j = 1$.

The X_i's, where $i = 1 \ldots N$, are the data vectors we are trying to cluster, and N is the number of samples. To cluster X_i, we need to estimate the parameters, Φ. One method for estimating Φ is to find Φ that maximizes the log-likelihood, $\log f(X|\Phi) = \sum_{i=1}^{N} \log f(X_i|\Phi)$. To compute $f(X_i|\Phi)$, we need to know the cluster (the missing data) to which X_i (the observed data) belongs. We apply the EM algorithm, which provides us with "soft-clustering" information; i.e., a data point X_i can belong to more than one cluster (weighted by its probability to belong to each cluster). The expectation-maximization (EM) algorithm, introduced in some generality by Dempster, Laird and Rubin in 1977, is an iterative approximation algorithm for computing the maximum likelihood (ML) estimate of missing data problems.

Going through the derivation of applying EM on our Gaussian mixture model, we obtain the following EM update equations (Wolfe, 1970):

$$E[z_{ij}]^{(t)} = p(z_{ij} = 1|X, \Phi^{(t)}) = \frac{f_j(X_i|\Phi_j^{(t)})\pi_j^{(t)}}{\sum_{s=1}^{k} f_s(X_i|\Phi_s^{(t)})\pi_s^{(t)}}; \tag{6}$$

$$\pi_j^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} E[z_{ij}]^{(t)}; \quad (7)$$

$$\mu_j^{(t+1)} = \frac{1}{N\pi_j^{(t+1)}}\sum_{i=1}^{N} E[z_{ij}]^{(t)} X_i; \quad (8)$$

$$\Sigma_j^{(t+1)} = \frac{1}{N\pi_j^{(t+1)}}\sum_{i=1}^{N} E[z_{ij}]^{(t)} (X_i-\mu_j^{(t+1)})(X_i-\mu_j^{(t+1)})^T; \quad (9)$$

where $E[z_{ij}]$ is the probability that X_i belongs to cluster j given our current parameters and X_i, $\sum_{i=1}^{N} E[z_{ij}]$ is the estimated number of data points in class j, and the superscript t refers to the iteration.

Appendix B. Additional Proofs on ML's Bias with Dimension

In this appendix, we prove Theorem 4.1 and Corollary 4.1 which state the condition that needs to be satisfied for the maximum likelihood of feature subset A, $ML(\Phi_A)$, to be greater than or equal to the maximum likelihood of feature subset B, $ML(\Phi_B)$. To prove these results, we first define the maximum likelihood criterion for a mixture of Gaussians, prove Lemma B.1 which derives a simplified form of $\exp(Q(\Phi,\Phi))$ for a finite Gaussian mixture, and Lemma B.2 which states the condition that needs to be satisfied for the complete expected data log-likelihood $Q(\cdot)$ function given the observed data and the parameter estimates in feature subset A, $Q(\Phi_A,\Phi_A)$, to be greater than or equal to the $Q(\cdot)$ function of feature subset B, $Q(\Phi_B,\Phi_B)$.

The maximum likelihood of our data, X, is

$$ML = \max_{\Phi}(f(X|\Phi)) = \max_{\Phi}\prod_{i=1}^{N}(\sum_{j=1}^{k}\pi_j f_j(X_i|\theta_j)), \quad (10)$$

where $f_j(X_i|\theta_j)$ is the probability density function for class j, π_j is the mixing proportion of class j (prior probability of class j), N is the number of data points, k is the number of clusters, X_i is a d-dimensional random data vector, θ_j is the set of parameters for class j, $\Phi=(\pi,\theta)$ is the set of all parameters and $f(X|\Phi)$ is the probability density function of our observed data $X=X_1,X_2,\ldots X_N$ given the parameters Φ. We choose the feature subset that maximizes this criterion.

Lemma B.1 *For a finite mixture of Gaussians,*

$$\exp(Q(\Phi,\Phi)) = \prod_{j=1}^{K}\pi_j^{N\pi_j}\frac{1}{(2\pi)^{\frac{dN\pi_j}{2}}|\Sigma_j|^{\frac{N\pi_j}{2}}}e^{-\frac{1}{2}dN\pi_j},$$

where x_i, $i=1\ldots N$, are the N observed data points, z_{ij} is the missing variable equal to one if x_i belongs to cluster j and zero otherwise, π_j is the mixture proportion, μ_j is the mean and Σ_j is the covariance matrix of each Gaussian cluster respectively, and $\Phi=(\pi,\mu,\Sigma)$ is the set of all estimated parameters.

Proof:

$$\begin{aligned}
Q(\Phi,\Phi) &\stackrel{\triangle}{=} E_{z|x}[\log f(x,z|\Phi)|x,\Phi] \\
&= E_{z|x}[\log f(x|z,\Phi)|x,\Phi] + E_{z|x}[\log f(z|\Phi)|x,\Phi] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{K} p(z_{ij}=1|x,\Phi)\log f_j(x_i|\phi_j) + \sum_{i=1}^{N}\sum_{j=1}^{K} p(z_{ij}=1|x,\Phi)\log\pi_j \\
&= \sum_{i=1}^{N}\sum_{j=1}^{K} p(z_{ij}=1|x,\Phi)\log(\pi_j f_j(x_i|\phi_j)) \qquad (11) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{K} E[z_{ij}]\log(\pi_j f_j(x_i|\phi_j)) \\
\exp(Q(\Phi,\Phi)) &= \prod_{i=1}^{N}\prod_{j=1}^{K}(\pi_j f_j(x_i|\phi_j))^{E[z_{ij}]}. \qquad (12)
\end{aligned}$$

Substituting our parameter estimates to Equation 12 and sample data x_i's,

$$\begin{aligned}
\exp(Q(\Phi,\Phi)) &= \prod_{j=1}^{K}\pi_j^{\sum_{i=1}^{N}E[z_{ij}]}\prod_{i=1}^{N}\Big(\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_j|^{\frac{1}{2}}}e^{-\frac{1}{2}(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)}\Big)^{E[z_{ij}]} \\
&= \prod_{j=1}^{K}\pi_j^{N\pi_j}\frac{1}{(2\pi)^{\frac{dN\pi_j}{2}}|\Sigma_j|^{\frac{N\pi_j}{2}}}e^{-\frac{1}{2}\sum_{i=1}^{N}E[z_{ij}](x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)}. \qquad (13)
\end{aligned}$$

Simplifying the exponent of e we obtain

$$\begin{aligned}
& -\frac{1}{2}\sum_{i=1}^{N}E[z_{ij}](x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j) \\
=\ & -\frac{1}{2}\sum_{i=1}^{N}E[z_{ij}]tr((x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)) \\
=\ & -\frac{1}{2}\sum_{i=1}^{N}E[z_{ij}]tr(\Sigma_j^{-1}(x_i-\mu_j)(x_i-\mu_j)^T) \\
=\ & -\frac{1}{2}tr(\Sigma_j^{-1}(\sum_{i=1}^{N}E[z_{ij}](x_i-\mu_j)(x_i-\mu_j)^T)).
\end{aligned}$$

Adding and subtracting $\overline{x_j}$, where $\overline{x_j} = \frac{1}{N\pi_j}\sum_{i=1}^{N}E[z_{ij}]x_i$, this last expression becomes

$$-\frac{1}{2}tr(\Sigma_j^{-1}(\sum_{i=1}^{N}E[z_{ij}](x_i-\overline{x_j}+\overline{x_j}-\mu_j)(x_i-\overline{x_j}+\overline{x_j}-\mu_j)^T)).$$

Cancelling cross-product terms yields

$$-\frac{1}{2}tr(\Sigma_j^{-1}(\sum_{i=1}^{N}E[z_{ij}](x_i-\overline{x_j})(x_i-\overline{x_j})^T + \sum_{i=1}^{N}E[z_{ij}](\overline{x_j}-\mu_j)(\overline{x_j}-\mu_j)^T)),$$

and finally substituting the parameter estimates (Equations 7-9) gives the expression

$$\begin{aligned} & -\frac{1}{2} tr(\Sigma_j^{-1} \Sigma_j N\pi_j) \\ = \quad & -\frac{1}{2} dN\pi_j. \end{aligned} \tag{14}$$

Thus, $\exp(Q(\Phi,\Phi))$ can be expressed as

$$\exp(Q(\Phi,\Phi)) \quad = \quad \prod_{j=1}^{K} \pi_j^{N\pi_j} \frac{1}{(2\pi)^{\frac{dN\pi_j}{2}} |\Sigma_j|^{\frac{N\pi_j}{2}}} e^{-\frac{1}{2} dN\pi_j} \qquad \square$$

Lemma B.2 *Assuming identical clustering assignments for feature subsets A and B with dimensions $d_B \geq d_A$, $Q(\Phi_A,\Phi_A) \geq Q(\Phi_B,\Phi_B)$ iff*

$$\prod_{j=1}^{k} \left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j} \right)^{\pi_j} \quad \geq \quad \frac{1}{(2\pi e)^{(d_B - d_A)}}.$$

Proof:

Applying Lemma B.1, and assuming subsets A and B have equal clustering assignments,

$$\begin{aligned} \frac{\exp(Q(\Phi_A,\Phi_A))}{\exp(Q(\Phi_B,\Phi_B))} \quad & \geq \quad 1; \\ \frac{\prod_{j=1}^{k} \pi_j^{N\pi_j} \frac{1}{(2\pi e)^{\frac{d_A N\pi_j}{2}}} \frac{1}{|\Sigma_A|_j^{\frac{N\pi_j}{2}}}}{\prod_{j=1}^{k} \pi_j^{N\pi_j} \frac{1}{(2\pi e)^{\frac{(d_B) N\pi_j}{2}}} \frac{1}{|\Sigma_B|_j^{\frac{N\pi_j}{2}}}} \quad & \geq \quad 1. \end{aligned} \tag{15}$$

Given $d_B \geq d_A$, without loss of generality and cancelling common terms,

$$\begin{aligned} \prod_{j=1}^{k} \left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j} \right)^{\frac{N\pi_j}{2}} (2\pi e)^{\frac{(d_B - d_A) N\pi_j}{2}} \quad & \geq \quad 1; \\ \prod_{j=1}^{k} \left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j} \right)^{\pi_j} (2\pi e)^{(d_B - d_A)\pi_j} \quad & \geq \quad 1; \\ (2\pi e)^{(d_B - d_A) \sum_{j=1}^{k} \pi_j} \prod_{j=1}^{k} \left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j} \right)^{\pi_j} \quad & \geq \quad 1; \\ \prod_{j=1}^{k} \left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j} \right)^{\pi_j} \quad & \geq \quad \frac{1}{(2\pi e)^{(d_B - d_A)}}. \qquad \square \end{aligned}$$

Theorem B.1 (Theorem 4.1 restated) *For a finite multivariate Gaussian mixture, assuming identical clustering assignments for feature subsets A and B with dimensions $d_B \geq d_A$, $ML(\Phi_A) \geq ML(\Phi_B)$ iff*

$$\prod_{j=1}^{k} \left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j} \right)^{\pi_j} \quad \geq \quad \frac{1}{(2\pi e)^{(d_B - d_A)}}.$$

Proof:
The log-likelihood, $\log L(\Phi') = \log f(x|\Phi')$.

$$\begin{aligned} \log L(\Phi') &= E_{z|x}[\log f(x,z|\Phi')|x,\Phi] - E_{z|x}[\log f(z|x,\Phi')|x,\Phi] \\ &\stackrel{\triangle}{=} Q(\Phi',\Phi) + H(\Phi',\Phi). \end{aligned}$$

$$\begin{aligned} H(\Phi,\Phi) &= -E[\log f(z|x,\Phi)|x,\Phi] \\ &= -\sum_{i=1}^{N}\sum_{j=1}^{k} p(z_{ij}=1|x_i,\phi_j)\log p(z_{ij}=1|x_i,\phi_j) \\ &= -\sum_{i=1}^{N}\sum_{j=1}^{k} E[z_{ij}]\log E[z_{ij}]. \end{aligned} \tag{16}$$

Since the identical clustering assignment assumption means that $E[z_{ij}]$ for feature set A is equal to $E[z_{ij}]$ for feature set B,

$$H(\Phi_A,\Phi_A) = H(\Phi_B,\Phi_B).$$

Thus,

$$\frac{ML(\Phi_A)}{ML(\Phi_B)} = \frac{\exp(Q(\Phi_A,\Phi_A))}{\exp(Q(\Phi_B,\Phi_B))}.$$

For a finite Gaussian mixture, from Lemma B.2, $\frac{ML(\Phi_A)}{ML(\Phi_B)} \geq 1$ iff

$$\prod_{j=1}^{k}\left(\frac{|\Sigma_B|_j}{|\Sigma_A|_j}\right)^{\pi_j} \geq \frac{1}{(2\pi e)^{(d_B-d_A)}}. \qquad \square$$

Corollary B.1 (Corollary 4.1 restated) *For a finite multivariate Gaussian mixture, assuming identical clustering assignments for feature subsets X and (X,Y), where X and Y are disjoint, $ML(\Phi_X) \geq ML(\Phi_{XY})$ iff*

$$\prod_{j=1}^{k}|\Sigma_{YY}-\Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}|_j^{\pi_j} \geq \frac{1}{(2\pi e)^{d_Y}}.$$

Proof:

Applying Theorem 4.1, and if we let A be the marginal feature vector X with dimension d_X and B be the joint feature vector (X,Y) with dimension d_X+d_Y (where subsets X and Y are disjoint), then the maximum likelihood of X is greater than or equal to the maximum likelihood of (X,Y) iff

$$\begin{aligned} \frac{ML(\Phi_X)}{ML(\Phi_{XY})} &\geq 1 \\ (2\pi e)^{d_Y}\prod_{j=1}^{k}\left(\frac{\left|\begin{array}{cc}\Sigma_{XX} & \Sigma_{XY}\\ \Sigma_{YX} & \Sigma_{YY}\end{array}\right|_j}{|\Sigma_{XX}|_j}\right)^{\pi_j} &\geq 1. \end{aligned}$$

Exercise 4.11 of Johnson and Wichern (1998) shows that for any square matrix A,

$$
\begin{aligned}
A &= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \\
|A| &= |A_{22}||A_{11} - A_{12}A_{22}^{-1}A_{21}| \quad \text{for } |A_{22}| \neq 0 \\
&= |A_{11}||A_{22} - A_{21}A_{11}^{-1}A_{12}| \quad \text{for } |A_{11}| \neq 0.
\end{aligned}
$$

Thus,

$$
\begin{aligned}
(2\pi e)^{d_Y} \prod_{j=1}^{k} \left(\frac{1}{|\Sigma_{XX}|_j} |\Sigma_{XX}|_j |\Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}|_j \right)^{\pi_j} &\geq 1 \\
\prod_{j=1}^{k} |\Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}|_j^{\pi_j} &\geq \frac{1}{(2\pi e)^{d_Y}}. \qquad \square
\end{aligned}
$$

Now, what do these results mean? One can compute the maximum log-likelihood, log ML efficiently as $Q(\Phi,\Phi) + H(\Phi,\Phi)$ by applying Lemma B.1 and Equation 16. Lemma B.1 shows that the ML criterion prefers low covariance clusters. Equation 16 shows that the ML criterion penalizes increase in cluster entropy. Theorem 4.1 and Corollary 4.1 reveal the dependencies of comparing the ML criterion for different dimensions. Note that the left hand side term of Corollary 4.1 is the determinant of the covariance of $f(Y|X)$. It is the covariance of Y minus the correlation of Y and X. For a criterion measure to be unbiased with respect to dimension, the criterion value should be the same for the different subsets when the cluster assignments are equal (and should not be dependent on the dimension). But, in this case, ML increases when the additional feature has small variance and decreases when the additional feature has large variance.

References

R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 94–105, Seattle, WA, June 1998. ACM Press.

D. W. Aha and R. L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the 1994 AAAI Workshop on Case-Based Reasoning*, pages 106–112, Seattle, WA, 1994. AAAI Press.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6):716–723, December 1974.

H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552, Anaheim, CA, 1991. AAAI Press.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases. In *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, 1998.

C. A. Bouman, M. Shapiro, G. W. Cook, C. B. Atkins, and H. Cheng. Cluster: An unsupervised algorithm for modeling gaussian mixtures. In *http://dynamo.ecn.purdue.edu/∼bouman/software/cluster*, October 1998.

P. S. Bradley and U. M. Fayyad. Refining initial points for K-Means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99, San Francisco, CA, 1998. Morgan Kaufmann.

L. Breiman. Wald lecture II: Looking inside the black box, 2002. 277th meeting of the Institute of Mathematical Statistics, Banff, Alberta, Canada. http://stat-www.berkeley.edu/users/breiman/wald2002-2.pdf.

C. Cardie. Using decision trees to improve case-based learning. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 25–32, Amherst, MA, 1993. Morgan Kaufmann.

R. Caruana and D. Freitag. Greedy attribute selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36, New Brunswick, NJ, 1994. Morgan Kaufmann.

W. C. Chang. On using principal components before separating a mixture of two multivariate normal distributions. *Applied Statistics*, 32:267–275, 1983.

P. Cheeseman and J. Stutz. Bayesian classification (autoclass): theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180, Cambridge, MA, 1996. AAAI/MIT Press.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal Royal Statistical Society, Series B*, 39(1):1–38, 1977.

M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 92–97, Nashville, TN, 1997. Morgan Kaufmann.

J. Doak. An evaluation of feature selection methods and their application to computer security. Technical report, University of California at Davis, 1992.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, NY, 1973.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (Second Edition)*. Wiley & Sons, NY, 2001.

J. G. Dy and C. E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 247–254, Stanford University, 2000a. Morgan Kaufmann.

J. G. Dy and C. E. Brodley. Interactive visualization and feature selection for unsupervised data. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 360–364, Boston, MA, August 2000b. ACM Press.

J. G. Dy and C. E. Brodley. TR-CDSP-03-53: feature selection for unsupervised learning. Technical report, Northeastern University, December 2003.

J. G. Dy, C. E. Brodley, A. Kak, L. S. Broderick, and A. M. Aisen. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):373–378, March 2003.

J. G. Dy, C. E. Brodley, A. Kak, C. R. Shyu, and L. S. Broderick. The customized-queries approach to CBIR using EM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 400–406, Fort Collins, CO, June 1999. IEEE Computer Society Press.

U. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 194–198, New York, August 1998. AAAI Press.

M. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:381–396, 2002.

D. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, 4:147–179, 1996.

D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2 (2):139–172, 1987.

E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21:768, 1965.

C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. Technical Report Technical Report No. 380, University of Washington, Seattle, WA, 2000.

J. H. Friedman. Exploratory projection pursuit. *Journal American Statistical Association*, 82:249–266, 1987.

J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *to appear at Journal Royal Statistical Society*, 2003.

K. Fukunaga. *Statistical Pattern Recognition (second edition)*. Academic Press, San Diego, CA, 1990.

J. H. Gennari. Concept formation and attention. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 724–728, Chicago, IL, 1991. Lawrence Erlbaum.

J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–61, 1989.

Z. Ghahramani and M. J. Beal. Variational inference for bayesian mixtures of factor analysers. In S. A. Solla, T. K. Leen, and K. Muller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 449–455. MIT Press, 2000.

Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report Technical Report CRG-TR-96-1, University of Toronto, Toronto, Canada, 1996.

J. A. Hartigan. Statistical theory in clustering. *Journal of Classification*, 2:63–76, 1985.

P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann.

R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, 4 edition, 1998.

N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9:1493–1516, 1997.

Y. S. Kim, N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6:531–556, 2002.

K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the Ninth International Workshop on Machine Learning*, pages 249–256, Aberdeen, Scotland, 1992. Morgan Kaufmann.

J. Kittler. Feature set search algorithms. In *Pattern Recognition and Signal Processing*, pages 41–60, 1978.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

I. Kononenko. Estimating attributes: analysis and extensions of relief. In *Proceedings of the European Conference on Machine Learning*, 1994.

P. Langley and S. Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 113–117, Seattle, 1994. AAAI Press.

M. H. Law, M. Figueiredo, and A. K. Jain. Feature selection in mixture-based clustering. In *Advances in Neural Information Processing Systems 15*, Vancouver, December 2002.

H. Liu and R. Setiono. Dimensionality reduction via discretization. *Knowledge-Based Systems*, 9 (1):67–72, February 1996.

T. Marill and D. M. Green. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9:11–17, 1963.

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley, New York, 1997.

G. J. McLachlan and K. E. Basford. *Mixture Models, Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

G. Milligan. A monte carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199, June 1981.

B. Mirkin. Concept learning and feature selection based on square-error clustering. *Machine Learning*, 35:25–39, 1999.

T. K. Moon. The expectation-maximization algorithm. In *IEEE Signal Processing Magazine*, pages 47–59, November 1996.

P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–922, September 1977.

M. J. Pazzani. Searching for dependencies in bayesian classifiers. In *Proceedings of the fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, Florida, 1995.

D. Pelleg and A. Moore. *X*-means: Extending *K*-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734. Morgan Kaufmann, San Francisco, CA, 2000.

J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

S. J. Russell and P. Norvig. *Artificial Intelligence a Modern Approach*. Prentice Hall, Saddle River, NJ, 1995.

G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

M. Singh and G. M. Provan. A comparison of induction algorithms for selective and non-selective bayesian classifiers. In *Machine Learning: Proceedings of the Twelfth International Conference*, pages 497–505, 1995.

P. Smyth. Clustering using Monte Carlo cross-validation. In E. Simoudis, J. Han, and U. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 126–133, Portland, OR, 1996. AAAI Press.

L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 389–397, Bled, Slovenia, 1999. Morgan Kaufmann.

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. John Wiley and Sons, Chichester, UK, 1985.

N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. In *Neural Computation*, 1999. To appear.

S. Vaithyanathan and B. Dom. Model selection in unsupervised learning with applications to document clustering. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 433–443, Bled, Slovenia, 1999. Morgan Kaufmann.

S. Vaithyanathan and B. Dom. Hierarchical unsupervised learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1039–1046, Palo Alto, CA, 2000. Morgan Kaufmann.

J. H. Wolfe. Pattern clustering by multivariate mixture analysis. *Multivariate Behavioral Research*, 5(3):101–116, 1970.

C. F. J. Wu. On the convergence properties of the em algorithm. *The Annals of Statistics*, 11(1): 95–103, 1983.

Journal of Machine Learning Research 5 (2004) 891–912 Submitted 3/04; Published 8/04

Some Dichotomy Theorems for Neural Learning Problems

Michael Schmitt MSCHMITT@LMI.RUHR-UNI-BOCHUM.DE
Lehrstuhl Mathematik und Informatik
Fakultät für Mathematik
Ruhr-Universität Bochum
D–44780 Bochum, Germany

Editor: Manfred K. Warmuth

Abstract

The computational complexity of learning from binary examples is investigated for linear threshold neurons. We introduce combinatorial measures that create classes of infinitely many learning problems with sample restrictions. We analyze how the complexity of these problems depends on the values for the measures. The results are established as dichotomy theorems showing that each problem is either NP-complete or solvable in polynomial time. In particular, we consider consistency and maximum consistency problems for neurons with binary weights, and maximum consistency problems for neurons with arbitrary weights. We determine for each problem class the dividing line between the NP-complete and polynomial-time solvable problems. Moreover, all efficiently solvable problems are shown to have constructive algorithms that require no more than linear time on a random access machine model. Similar dichotomies are exhibited for neurons with bounded threshold. The results demonstrate on the one hand that the consideration of sample constraints can lead to the discovery of new efficient algorithms for non-trivial learning problems. On the other hand, hard learning problems may remain intractable even for severely restricted samples.

Keywords: linear threshold neuron, consistency problem, computational complexity, linear-time algorithm, NP-completeness

1. Introduction

The ability to learn is certainly one of the most challenging qualities that people have ever demanded from computers. There is no doubt that a major advancement in the development of learning algorithms has been made due to the use of neural networks. Today, they provide the basis for some of the most successful learning techniques. This achievement, however, is compromised by the fact that the running times required by neural learning algorithms are often immense. On the theoretical side, investigations of the computational complexity of learning problems have supported the conjecture that neural learning is generally hard: Theory considers an algorithm as efficient if its running time is bounded by a polynomial in the input length. Based on the theory of NP-completeness (see, e.g., Garey and Johnson, 1979) researchers have established numerous intractability results for neural learning problems. Consequently, no efficient, that is, polynomial-time, algorithm for solving these problems can exist if the complexity classes P and NP are different. (See, e.g., Šíma, 2002, for a comprehensive list of hardness results concerning single neurons and neural networks.)

A useful approach for studying the complexity of learning is to consider the consistency problem. Associated with a class of functions, the so-called hypothesis class, the consistency problem is a decision problem. It poses the question whether for a given set of labeled examples, the training

sample, there is some hypothesis in the class that is consistent with all examples, that is, assigns the correct output values. The consistency problem is also known as loading problem (Judd, 1990), fitting problem (Natarajan, 1991), or training problem (Blum and Rivest, 1992). A variant of the consistency problem is the maximum consistency problem, also known as minimizing disagreement problem. Its instances consist of a training sample and a natural number k. The question is to decide whether there is a hypothesis consistent with some subset of at least k examples. The maximum consistency problem is a generalization of the consistency problem in that the latter is a subproblem of the former: The consistency problem can be obtained from the maximum consistency problem by setting k equal to the number of examples. Consequently, if for a given hypothesis class the consistency problem is NP-hard then the maximum consistency problem is NP-hard, too. Consistency and maximum consistency problems are studied as key problems in the computational models of learning known as probably approximately correct (PAC) learning and agnostic PAC learning. A fundamental result states that, under the assumption that the complexity classes RP and NP are different, PAC learning cannot be efficient for a hypothesis class if the consistency problem for this class is NP-hard (Pitt and Valiant, 1988; Blumer et al., 1989). An analogous result links the model of agnostic PAC learning with the maximum consistency problem (Kearns et al., 1994; Anthony and Bartlett, 1999).

There has been perseverating interest in theoretical issues of learning using single neurons as hypothesis class (see, e.g., Servedio, 2002, and the references there). One of the extensively studied models is the linear threshold neuron, also known as McCulloch-Pitts neuron (McCulloch and Pitts, 1943). It serves as elementary building block for many neural network types (see, e.g., Haykin, 1999). While the linear threshold neuron has arbitrary real-valued weights, a variant that has gained particular attention is obtained by restricting the weights to binary values (see, e.g., Golea and Marchand, 1993; Fang and Venkatesh, 1996; Kim and Roche, 1998; Sommer and Palm, 1999). With binary weights a linear threshold neuron is still able to compute fundamental Boolean functions such as conjunction, disjunction, and r-of-k threshold functions[1] that have been of specific interest in questions of learning (Hampson and Volper, 1986; Littlestone, 1988; Pitt and Valiant, 1988).

In this article, we investigate the complexity of consistency and maximum consistency problems for linear threshold neurons with binary and with arbitrary weights. We do this under the assumption that the training examples are binary, that is, have entries from $\{0,1\}$. While the consistency problem for linear threshold neurons with arbitrary weights is known to be solvable in polynomial time using methods of linear programming (see, e.g., Maass and Turán, 1994; Anthony and Bartlett, 1999) we focus on those problems that are NP-complete: The consistency problem for linear threshold neurons with binary weights has been proved to be NP-complete by Pitt and Valiant (1988). Hence, also the maximum consistency problem for these neurons is NP-complete. The maximum consistency problem for linear threshold neurons with arbitrary weights is known to be NP-complete from the work of Höffgen et al. (1995) improving an earlier result by Amaldi (1991).

We take a closer look at these problems by introducing and analyzing subproblems. In other words, we raise the question whether these problems become easier when the training samples are restricted. To this end we define two combinatorial measures for characterizing the complexity of a sample. The first quantity, called coincidence, indicates the maximum number of components in which any two examples are both non-zero. The second measure, called sparseness, represents the maximum number of non-zero components, or the Hamming-weight, of any example. The

1. A Boolean r-of-k threshold function outputs 1 if and only if at least r in a specified subset of k variables have the value 1.

choice of these measures is guided by the insight that learning for single neurons is easy when the coincidence or the sparseness of the samples is severely limited. So, it is not hard to see that the consistency problem is trivial when the examples are pairwise orthogonal, that is, when the sample has coincidence zero. The same observation can be made for the case when the examples are required to be extremely sparse. A further motivation for introducing sparseness arises from investigations of neural associative memories. In these, sparseness has been shown to be relevant for their storage capacities using specific learning rules (Palm, 1980).

By introducing coincidence and sparseness as sample restrictions we obtain infinite classes of consistency and maximum consistency problems: For each pair c,d of natural numbers there is the (maximum) consistency problem with coincidence c and sparseness d. The question arises how the complexity of the consistency and the maximum consistency problems depends on the values for coincidence and sparseness. As one of the main results it emerges that these problems are already NP-complete when coincidence and sparseness are bounded, a fact that does not follow from previous work. Moreover, we give a complete categorization of these consistency and maximum consistency problems into NP-complete and polynomial-time solvable. In other words, we establish so-called dichotomy theorems for these problems. Given an infinite class of problems, a dichotomy theorem states that each problem is either NP-complete or contained in the complexity class P. Thus, if P $\neq$ NP, the dichotomy theorem separates the efficiently solvable problems from the intractable ones. A dichotomy theorem is an important finding since it is known that if P and NP are different then there are infinitely many problems in NP that are neither in P nor NP-complete (see, e.g., Garey and Johnson, 1979). It is therefore significant that none of these problems of intermediate complexity is found among the (maximum) consistency problems considered here. In computational complexity there are only a few dichotomy theorems known. The most popular one is that for k-SAT which states that the satisfiability problem for conjunctions of Boolean clauses, where each clause has size at most k, is NP-complete when k is at least 3 and solvable in polynomial time for k at most 2. The dichotomy of k-SAT emerges as a special case of the renowned result of Schaefer (1978) who established a more general dichotomy theorem for satisfiability problems. Concerning more recent results we refer to Kirousis and Kolaitis (2003) where also a brief survey on dichotomy theorems in computational complexity is given. The first dichotomy theorem in conjunction with learning problems has been provided by Dalmau (1999) for learning quantified Boolean formulas (see also Dalmau and Jeavons, 2003).

In detail, we prove the following dichotomy theorems: The consistency problem for the linear threshold neuron with binary weights is NP-complete if the samples have coincidence at least 1 and sparseness at least 4. On the other hand, if the samples have coincidence 0 or sparseness at most 3, the problem becomes solvable in polynomial time. Further, the maximum consistency problem for the linear threshold neuron with binary weights is NP-complete for coincidence at least 1 and sparseness at least 2, whereas for coincidence 0 or sparseness 1 it is in P. This last dichotomy theorem holds also for the maximum consistency problem with respect to the linear threshold neuron with arbitrary weights. All polynomial-time results are established by showing that the problems can be solved in linear time on a random access machine (RAM) model. Moreover, the algorithms we present for these problems are constructive, that is, they calculate a solution if one exists. We further obtain dichotomies for linear threshold neurons with binary weights and a bounded threshold. In particular, the dichotomy of the consistency problems that holds for these neurons with arbitrary threshold is also existent when the threshold is bounded by some value larger than or equal to 2. On the other hand, if the bound on the threshold is at most 1, the consistency problem is solvable

in linear time on a RAM. For the maximum consistency problem and linear threshold neurons with binary weights NP-completeness is found at a value for the threshold of 1 or greater, whereas linear-time solvability holds when the threshold is equal to 0.

The dichotomy theorems presented here extend the NP-completeness results for learning with single neurons to an infinite class of subproblems. Therefore, they confirm that neural learning belongs to the hardest computational problems that one needs to solve in practice. On the other hand, those problems that we find to be solvable in polynomial time can be solved even in linear time on a RAM. The algorithms designed for these problems give interesting new insights. Moreover, efficiently solvable subproblems have been successfully used for the design of improved heuristics for NP-complete problems (Zheng and Stuckey, 2002). Therefore, it is reasonable to expect that the new linear-time algorithms presented here may lead to new and faster heuristics for learning in single neurons and neural networks.

In Section 2 we introduce the definitions of the basic concepts. The dichotomy theorems are stated in Section 3. The proofs are given in the following two sections: Section 4 is concerned with the NP-completeness results while the linear-time algorithms are presented in Section 5. Section 6 is devoted to some conclusions and open questions. In the appendix we establish the NP-completeness of a new variant of the satisfiability problem that is used in Section 4. We assume that the reader is familiar with the theory of NP-completeness as covered, for instance, by Garey and Johnson (1979).

Bibliographic Note. Some of the results in this article have been previously described in a technical report (Schmitt, 1994a) and presented at the International Conference on Artificial Neural Networks, ICANN '95, in Paris (Schmitt, 1995). This article is a thoroughly revised and extended version of these papers.

2. Neurons, Samples, Problems, and Algorithms

We begin by defining the neuron model and the function classes associated with it. Then we introduce the notion of a sample and specify the computational problems involved with neurons and samples. Finally, with regard to the linear-time algorithms presented in Section 5, we describe the computer model of the random access machine (RAM) on which the time bounds rely.

2.1 Neurons

A linear threshold neuron has parameters $w_1, \ldots, w_n, t \in \mathbb{R}$, where $w_1, \ldots, w_n$ are the weights and t is the threshold. The class of linearly separable Boolean functions in n variables, denoted $\mathcal{L}_n$, is the class of functions $f : \{0,1\}^n \to \{0,1\}$ that can be computed by a linear threshold neuron. Specifically, for every $f \in \mathcal{L}_n$ there are weights $w_1, \ldots, w_n$ and a threshold t such that for all $x \in \{0,1\}^n$,

$$f(x) \quad = \quad \begin{cases} 1 & \text{if } w_1 x_1 + \cdots + w_n x_n \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

We use $\mathcal{B}_n$ to denote the subclass of linearly separable Boolean functions in n variables that can be computed by a linear threshold neuron using only binary weights, that is, $w_1, \ldots, w_n \in \{0,1\}$. Further, $\mathcal{B}_n^\ell$ is the subclass of functions in $\mathcal{B}_n$ that can be computed with threshold $t \leq \ell$. Clearly, a value of $\ell \in \{0, \ldots, n\}$ is sufficient and we have $\mathcal{B}_n^n = \mathcal{B}_n$.

By omitting the subscript n we refer to the union of the respective classes such as, for instance, $\mathcal{L} = \bigcup_{n \geq 1} \mathcal{L}_n$.

2.2 Samples

A sample S is a finite set $S \subseteq \{0,1\}^n \times \{0,1\}$. The elements of S are called examples. Given a sample S, the set of positive examples $\mathrm{Pos}(S)$ and the set of negative examples $\mathrm{Neg}(S)$ of S are defined by

$$\begin{aligned} \mathrm{Pos}(S) &= \{x \mid (x,1) \in S\}, \\ \mathrm{Neg}(S) &= \{x \mid (x,0) \in S\}. \end{aligned}$$

A sample may contain examples that are both positive and negative. Otherwise, that is, if $\mathrm{Pos}(S) \cap \mathrm{Neg}(S) = \emptyset$, the sample is said to be consistent.[2] The domain of a sample S, denoted $\mathrm{Dom}(S)$, is the set

$$\mathrm{Dom}(S) = \mathrm{Pos}(S) \cup \mathrm{Neg}(S).$$

We introduce two combinatorial measures that assign natural numbers to samples: coincidence and sparseness. The coincidence of a sample S, denoted $\mathrm{Coin}(S)$, is

$$\mathrm{Coin}(S) = \begin{cases} \max\{x \cdot y \mid x,y \in \mathrm{Dom}(S),\ x \neq y\} & \text{if } |S| \geq 2, \\ 0 & \text{otherwise,} \end{cases}$$

where $x \cdot y$ denotes the inner product. The sparseness of S, $\mathrm{Sparse}(S)$, is

$$\mathrm{Sparse}(S) = \begin{cases} \max\{x \cdot x \mid x \in \mathrm{Dom}(S)\} & \text{if } S \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, $\mathrm{Coin}(S)$ is the maximum number of components in which two different elements in the domain of S have a 1 in common, and $\mathrm{Sparse}(S)$ is the maximum number of 1s, or the Hamming weight, of an element. Note that $\mathrm{Coin}(S) \leq \mathrm{Sparse}(S)$. Thus, bounding the sparseness of samples by some constant also bounds their coincidence by the same constant.

2.3 Consistency Problems

Given a sample S, a function f is said to be consistent with S if f satisfies $f(x) = a$ for all $(x,a) \in S$. The consistency problem for a class $\mathcal{F}$ of Boolean functions is the problem to decide whether for an input sample S there exists some function $f \in \mathcal{F}$ that is consistent with S. Loading problem, fitting problem, and training problem are synonyms for the consistency problem.

Let $\mathcal{F}$ be a class of Boolean functions and let $c,d \in \mathbb{N}$. We define the following consistency problems for $\mathcal{F}$ with sample restrictions:

$\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c
Instance: A sample S with $\mathrm{Coin}(S) \leq c$.
Question: Is there a function $f \in \mathcal{F}$ that is consistent with S?

$\mathcal{F}$-CONSISTENCY WITH SPARSENESS d
Instance: A sample S with $\mathrm{Sparse}(S) \leq d$.
Question: Is there a function $f \in \mathcal{F}$ that is consistent with S?

2. The term "consistent sample" uses the word "consistent" in a somewhat different sense than the rest of the article.

The definition of the problem $\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d is similar.

The instances of the maximum consistency problem for a class $\mathcal{F}$ of functions consist of a sample S and a number $k \in \mathbb{N}$. The question is whether there exists a subset of S with at least k elements and a function $f \in \mathcal{F}$ consistent with that subset. Given numbers $c, d \in \mathbb{N}$, we define maximum consistency problems for $\mathcal{F}$ with sample restrictions as follows:

MAX-$\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c
Instance: A sample S with $\mathrm{Coin}(S) \leq c$ and a natural number k.
Question: Is there a subsample $S' \subseteq S$ with $|S'| \geq k$ and a function $f \in \mathcal{F}$ that is consistent with S'?

MAX-$\mathcal{F}$-CONSISTENCY WITH SPARSENESS d
Instance: A sample S with $\mathrm{Sparse}(S) \leq d$ and a natural number k.
Question: Is there a subsample $S' \subseteq S$ with $|S'| \geq k$ and a function $f \in \mathcal{F}$ that is consistent with S'?

Further, we consider the problem called MAX-$\mathcal{F}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d, which is the problem where the instances satisfy both constraints.

There are some natural relationships among the complexities of these problems. Clearly, an algorithm that solves the consistency problem with sparseness d solves also every consistency problem with sparseness less than d for the same function class. Furthermore, if the consistency problem is shown to be NP-hard for some sparseness d then it follows that all consistency problems with sparseness larger than d for the same function class are NP-hard as well, since the reduction established for sparseness d is also valid for larger bounds on the sparseness.

Similar considerations can be made with regard to coincidence. Moreover, complexity results for consistency problems with bounded coincidence have consequences for the complexity of consistency problems with bounded sparseness, and vice versa. In particular, an algorithm that solves the consistency problem for coincidence c solves also the consistency problem for sparseness $c+1$ since every sample with sparseness at most $c+1$ has coincidence no more than c. (Note that coincidence is measured using non-identical pairs only.) Correspondingly, NP-hardness for sparseness d implies NP-hardness for coincidence $d-1$.

Analogous interdependences hold among the maximum consistency problems with sample restrictions. Additionally, maximum consistency problems are related to consistency problems in that the consistency problem is a subproblem of the corresponding maximum consistency problem. The former is obtained from the latter by letting the number k be equal to the cardinality of the sample.

2.4 Linear-Time Algorithms

We give linear-time algorithms for all problems not shown to be NP-complete. While the complexity class P is known to be widely machine independent, this is not the case for the class of problems solved in linear time on some machine. In fact, a detailed account of the computer model must be given. We adopt the model of a random access machine (RAM) as it is the most realistic formal computer model and frequently used when analyzing the running time of algorithms (see, e.g., van Emde Boas, 1990). The instruction set of the RAM model comprises the standard load and store operations including the use of indirect addresses. Its arithmetic operations are addition and

subtraction, whereas multiplication and division are not available. We give time bounds for RAMs in the uniform measure, that is, the execution of an instruction counts as one unit of time. It is known that every RAM that requires time $t(n)$ in the uniform measure can be simulated on a multi-tape Turing machine in time $O(t^3(n))$ (Cook and Reckhow, 1973). In particular, a linear-time algorithm on a RAM ensures that the problem it solves belongs to P.

3. Dichotomy Theorems

We present five dichotomy theorems about the complexity of consistency problems with sample restrictions. They deal with the function classes $\mathcal{B}, \mathcal{B}^\ell$, and $\mathcal{L}$ introduced in Section 2.1 and the consistency problems and maximum consistency problems with sample restrictions defined in Section 2.3. The dichotomy theorems establish that, with one exception, each of these problems is either NP-complete or solvable in polynomial time. The latter is shown by exhibiting linear-time algorithms in all cases. The exception is the problem class arising from $\mathcal{L}$-CONSISTENCY. All subproblems of $\mathcal{L}$-CONSISTENCY can be solved in polynomial time as $\mathcal{L}$-CONSISTENCY itself is solvable in polynomial time using linear programming (see, e.g., Anthony and Bartlett, 1999; Maass and Turán, 1994). Therefore, if P $\neq$ NP, none of the subproblems of $\mathcal{L}$-CONSISTENCY is NP-complete.

The proofs for the statements that claim NP-completeness are presented in Section 4. The linear time bounds are established in Section 5 by giving algorithms that solve the problems constructively, that is, provide a solution if there exists one.

The first dichotomy theorem is concerned with the consistency problem for neurons with binary weights and arbitrary threshold.

Theorem 1 *$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d is NP-complete whenever $c \geq 1$ and $d \geq 4$. On the other hand, $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 and $\mathcal{B}$-CONSISTENCY WITH SPARSENESS d for $d \leq 3$ is solvable in linear time.*

An analogous result holds for neurons where the threshold is bounded by a constant of value at least 2. Moreover, if the threshold is required to be at most 1, we obtain solvability in linear time for all values of coincidence and sparseness.

Theorem 2 *For every $\ell \geq 2$, Theorem 1 holds with $\mathcal{B}^\ell$ in place of $\mathcal{B}$. On the other hand, $\mathcal{B}^1$-CONSISTENCY WITH COINCIDENCE c and $\mathcal{B}^1$-CONSISTENCY WITH SPARSENESS d is solvable in linear time for all c and d.*

These statements are complemented by the following two theorems that consider the corresponding maximum consistency problems.

Theorem 3 *MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d is NP-complete whenever $c \geq 1$ and $d \geq 2$. On the other hand, MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 and MAX-$\mathcal{B}$-CONSISTENCY WITH SPARSENESS d for $d \leq 1$ is solvable in linear time.*

A corresponding result for neurons with bounded threshold is the following one.

Theorem 4 *For every $\ell \geq 1$, Theorem 3 holds with $\mathcal{B}^\ell$ in place of $\mathcal{B}$. On the other hand, MAX-$\mathcal{B}^0$-CONSISTENCY WITH COINCIDENCE c and MAX-$\mathcal{B}^0$-CONSISTENCY WITH SPARSENESS d is solvable in linear time for all c and d.*

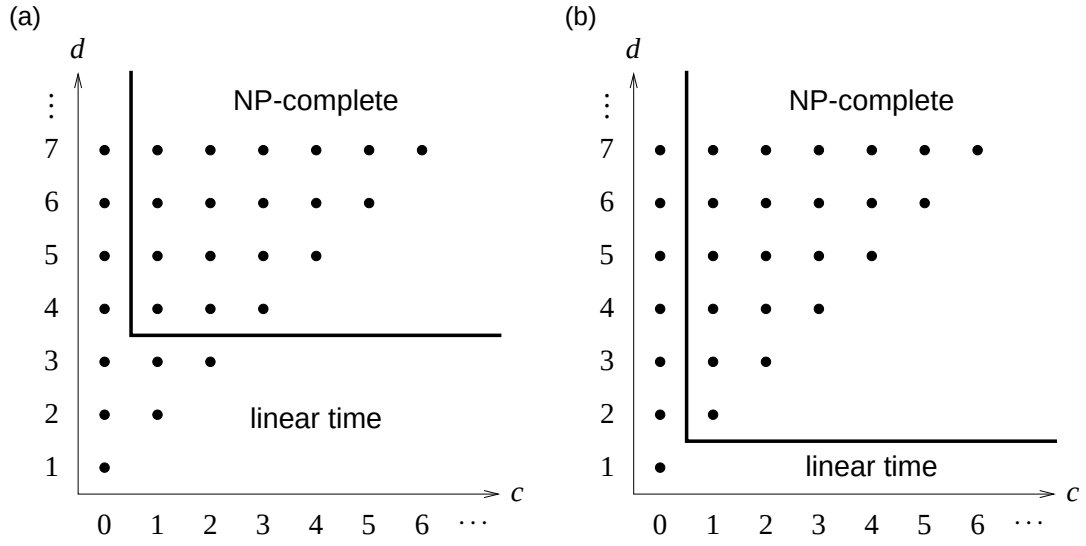


Figure 1: The dichotomies of the consistency and maximum consistency problems with coincidence c and sparseness d: **(a)** Consistency problems for the classes $\mathcal{B}$ and $\mathcal{B}^\ell$, where $\ell \geq 2$ (see Theorems 1 and 2); **(b)** maximum consistency problems for $\mathcal{B}, \mathcal{B}^\ell$, where $\ell \geq 1$, and $\mathcal{L}$ (see Theorems 3, 4, and 5). Linear time refers to the computing time required on a RAM model with uniform measure (see Section 2.4).

Finally, we present a dichotomy theorem for maximum consistency problems regarding neurons with arbitrary weights.

Theorem 5 MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE c AND SPARSENESS d *is NP-complete whenever* $c \geq 1$ *and* $d \geq 2$. *On the other hand,* MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{L}$-CONSISTENCY WITH SPARSENESS d *for* $d \leq 1$ *is solvable in linear time.*

The dichotomy theorems are illustrated in Figure 1. The dot with coordinates (c,d) represents the consistency problem or the maximum consistency problem, respectively, with coincidence c and sparseness d. Note that for $c \geq d$ the problem at coordinate (c,d) can be identified with the problem at $(d-1,d)$ (see the discussion at the end of Section 2.3). Problems with sparseness 0 are omitted as they are trivial.

4. NP-Complete Problems

We establish the NP-completeness for the consistency problems of Theorems 1 and 2 in Section 4.1. Section 4.2 deals with the maximum consistency problems of Theorems 3, 4, and 5. In all cases it is sufficient to prove NP-hardness since membership in NP is easy to derive: Guessing the weights

and the threshold and checking consistency or maximum consistency for the given function classes can be done in polynomial time.[3]

4.1 Consistency

The problem $\mathcal{B}$-CONSISTENCY without sample restrictions was shown to be NP-complete by Pitt and Valiant (1988). They have defined a reduction from the problem ZERO-ONE INTEGER PROGRAMMING that uses samples of unbounded coincidence and, hence, unbounded sparseness. To prove the stronger result, we make a new approach and establish a reduction from a problem that is a variant of the satisfiability problem which we call ALMOST DISJOINT POSITIVE 1-IN-3SAT.

ALMOST DISJOINT POSITIVE 1-IN-3SAT
Instance: A set U of variables, a collection $\mathcal{C}$ of subsets of U such that each subset $C \in \mathcal{C}$ has exactly three elements and each pair of subsets $C, D \in \mathcal{C}, C \neq D$, satisfies $|C \cap D| \leq 1$.
Question: Is there a truth assignment $\alpha : U \to \{0,1\}$ such that each subset in $\mathcal{C}$ has exactly one true variable?

The attribute "positive" refers to the fact that the variables are not negated; "almost disjoint" means that two subsets have at most one variable in common. The problem POSITIVE 1-IN-3SAT is known to be NP-complete (Garey and Johnson, 1979, p. 259). That the subproblem is NP-complete as well is claimed by the following statement. Its proof is given in the appendix.

Lemma 6 ALMOST DISJOINT POSITIVE 1-IN-3SAT *is NP-complete.*

This fact is employed in the following result. It covers the NP-completeness part of in Theorem 1. (Note that, as was argued in Section 2.3, NP-hardness of the consistency problem with coincidence c and sparseness d implies NP-hardness for any coincidence $c' \geq c$ and sparseness $d' \geq d$.)

Theorem 7 $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 4 *is NP-complete.*

Proof The proof is by reduction from the problem ALMOST DISJOINT POSITIVE 1-IN-3SAT defined above. Assume that an instance of this problem is given by the set of variables $U = \{u_1, \ldots, u_n\}$ and the collection of subsets $\mathcal{C} = \{c_1, \ldots, c_m\}$. The reduction maps this instance to a sample $S \subseteq \{0,1\}^{4n+m+2} \times \{0,1\}$. We introduce the following notation: Given a set $\{i_1, \ldots, i_j\} \subseteq \{1, \ldots, k\}$, let $[i_1, \ldots, i_j]_k$ denote that element of $\{0,1\}^k$ which has a 1 in positions $i_1, \ldots, i_j$, and 0 elsewhere. Then S is constructed as follows: For each variable $u_i \in U$, we define four examples

$$[i, n+i]_{4n+m+2} \in \mathrm{Neg}(S), \quad (1)$$

$$[2n+i, 3n+i]_{4n+m+2} \in \mathrm{Neg}(S), \quad (2)$$

$$[i, 3n+i, 4n+m+2]_{4n+m+2} \in \mathrm{Pos}(S), \quad (3)$$

$$[n+i, 2n+i, 4n+m+2]_{4n+m+2} \in \mathrm{Pos}(S). \quad (4)$$

3. For the function class $\mathcal{L}$ one makes use of the fact that on binary inputs a linear threshold neuron with arbitrary weights needs no more than polynomially in n many bits to represent weights and threshold (see, e.g., Schmitt, 1994b).

Position i	Value of w_i
$1,\ldots,n$	$\alpha(u_i)$
$n+1,\ldots,2n$	$1-\alpha(u_i)$
$2n+1,\ldots,3n$	$\alpha(u_i)$
$3n+1,\ldots,4n$	$1-\alpha(u_i)$
$4n+1,\ldots,4n+m$	1
$4n+m+1$	1
$4n+m+2$	1

Table 1: Definition of the weights for a given truth assignment α.

Each subset $c_\ell = \{u_i, u_j, u_k\} \in \mathcal{C}$ gives rise to three examples

$$[i, j, k, 4n+\ell]_{4n+m+2} \in \mathrm{Pos}(S), \tag{5}$$
$$[n+i, n+j, n+k]_{4n+m+2} \in \mathrm{Pos}(S), \tag{6}$$
$$[4n+\ell, 4n+m+2]_{4n+m+2} \in \mathrm{Pos}(S). \tag{7}$$

Finally, we add the three examples

$$[4n+m+1]_{4n+m+2} \in \mathrm{Neg}(S), \tag{8}$$
$$[4n+m+2]_{4n+m+2} \in \mathrm{Neg}(S), \tag{9}$$
$$[4n+m+1, 4n+m+2]_{4n+m+2} \in \mathrm{Pos}(S). \tag{10}$$

Obviously, S can be computed from U and $\mathcal{C}$ in polynomial time. Further, as can easily be checked, we have $\mathrm{Coin}(S) \leq 1$ and $\mathrm{Sparse}(S) \leq 4$. (Note that every pair of different subsets in $\mathcal{C}$ has at most one common variable. Hence, the examples in (5) and (6) that arise from different subsets have coincidence at most 1.)

It remains to show that the reduction is correct, that is, that $\mathcal{C}$ has a truth assignment with exactly one true variable in each subset if and only if there exists a function in $\mathcal{B}_{4n+m+2}$ that is consistent with S. In order to prove the "only if" part, assume that $\alpha : U \to \{0,1\}$ is a truth assignment that satisfies the assumption. Define the weights $w_1, \ldots, w_{4n}$ as follows: For $i = 1, \ldots, n$, let

$$\begin{aligned} w_i &= w_{2n+i} = \alpha(u_i), \\ w_{n+i} &= w_{3n+i} = 1-\alpha(u_i). \end{aligned}$$

The remaining weights and the threshold are determined by

$$w_{4n+1} = \cdots = w_{4n+m+2} = 1 \quad \text{and} \quad t = 2.$$

The representation of the truth assignment by the weights is shown in Table 1. Using the fact that α assigns exactly one 1 to each subset in $\mathcal{C}$, it is easy to verify that the function represented by these parameter values is consistent with S.

For establishing the "if" part, let the weights $w_1, \ldots, w_{4n+m+2} \in \{0,1\}$ and the threshold t represent a function that is consistent with S. The examples in (8), (9), and (10) imply that

$$\begin{aligned} w_{4n+m+1} &< t, \\ w_{4n+m+2} &< t, \\ w_{4n+m+1} + w_{4n+m+2} &\geq t, \end{aligned}$$

from which it follows that $w_{4n+m+1} = 1$ and $w_{4n+m+2} = 1$. Hence, the threshold satisfies $1 < t \leq 2$, and we may assume without loss of generality that $t = 2$. Then the examples in (7) entail that $w_{4n+1} = \cdots = w_{4n+m} = 1$. From the examples in (1) to (4) and the facts that $w_{4n+m+2} = 1$ and $t = 2$, we derive the inequalities

$$\begin{aligned} w_i + w_{n+i} &\leq 1, \\ w_{2n+i} + w_{3n+i} &\leq 1, \\ w_i + w_{3n+i} &\geq 1, \\ w_{n+i} + w_{2n+i} &\geq 1, \end{aligned}$$

for $i = 1, \ldots, n$. These imply the equalities

$$w_i + w_{n+i} = 1, \tag{11}$$

for $i = 1, \ldots, n$. We define the truth assignment $\alpha : U \to \{0,1\}$ by

$$\alpha(u_i) = w_i,$$

for $i = 1, \ldots, n$. The examples in (5), (6) and the fact that $t = 2$ yield the inequalities

$$\begin{aligned} w_i + w_j + w_k + w_{4n+\ell} &\geq 2, \\ w_{n+i} + w_{n+j} + w_{n+k} &\geq 2, \end{aligned}$$

for each subset $c_\ell = \{u_i, u_j, u_k\} \in \mathcal{C}$. From these, the fact that $w_{4n+1} = \cdots = w_{4n+m} = 1$, and the equations (11) we get that c_ℓ satisfies

$$\alpha(u_i) + \alpha(u_j) + \alpha(u_k) = 1.$$

Thus, each subset in $\mathcal{C}$ has exactly one 1 assigned by α. This completes the correctness proof for the reduction. ■

The foregoing proof shows that the same reduction can be used whenever the threshold is allowed to take on the value 2. Thus, it follows that the problems in Theorem 7 are NP-complete also for neurons with threshold at most ℓ, where $\ell \geq 2$. This observation establishes the NP-completeness statement of Theorem 2.

Corollary 8 *For every $\ell \geq 2$, $\mathcal{B}^\ell$-*CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 4 *is NP-complete.*

4.2 Maximum Consistency

Now, we address the NP-completeness of the maximum consistency problems. The following result covers the intractability part of Theorem 3.

Theorem 9 MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 2 *is NP-complete.*

Proof Höffgen et al. (1995) have shown that the maximum consistency problem for the class $\mathcal{L}$ and for samples without restrictions is NP-complete. They constructed a reduction from VERTEX COVER (Garey and Johnson, 1979, p. 190). This is the problem to decide, given a graph $G = (V,E)$ and a number k, whether there exists a set $V' \subseteq V$ with $|V'| \leq k$ such that for each edge $\{u,v\} \in E$ at least one of u and v belongs to V'. We use this problem for the reduction, too, but we take account of the fact that the weights are from the set $\{0,1\}$.

Let (V,E) and k be an instance of VERTEX COVER, where $V = \{v_1,\ldots,v_n\}$. We define the sample $S \subseteq \{0,1\}^{2n} \times \{0,1\}$ using the notation introduced in the proof of Theorem 7. For each $v_i \in V$, there is a "vertex example"

$$[i,n+i]_{2n} \quad \in \quad \mathrm{Neg}(S).$$

Each $e = \{v_i,v_j\} \in E$ results in two "edge examples"

$$\begin{aligned} [i,j]_{2n} &\in \mathrm{Pos}(S), \\ [n+i,n+j]_{2n} &\in \mathrm{Pos}(S). \end{aligned}$$

The cardinality of the subsample in the instance of the maximum consistency problem is set to the value $|S|-k$. It is obvious that $\mathrm{Coin}(S) \leq 1$ and $\mathrm{Sparse}(S) \leq 2$ holds, and that the reduction is computable in polynomial time.

We claim that (V,E) has a vertex cover of cardinality at most k if and only if there exists a function in $\mathcal{B}_{2n}$ that is consistent with at least $|S|-k$ examples. Let $V' \subseteq V$ be a vertex cover with at most k elements. Define the weights $w_1,\ldots,w_{2n}$ by

$$w_i \quad = \quad w_{n+i} \quad = \quad \begin{cases} 1 & \text{if } v_i \in V', \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 1,\ldots,n$, and let the threshold be $t = 1$. Then, the vertex example $[i,n+i]_{2n}$ is classified correctly if and only if $v_i \in V \setminus V'$. Further, since V' is a vertex cover, all edge examples are classified correctly. Thus, the function computed by this neuron is consistent with at least $|S|-k$ elements.

On the other hand, let the weights $w_1,\ldots,w_{2n}$ and the threshold t represent a function that is consistent with a subsample of S of cardinality at least $|S|-k$. Define a set V' of vertices as follows: For each vertex example that is classified wrongly include the corresponding vertex in V'; for each edge example that is classified wrongly arbitrarily choose one of the vertices the edge is incident with and include it in V'. Consequently, V' contains at most k elements. In order to show that V' is a vertex cover, assume that $\{v_i,v_j\} \in E$ is not covered, that is, we have $\{v_i,v_j\} \cap V' = \emptyset$. Then, by the construction of V', the function is consistent with the two vertex examples that arose from v_i and v_j, and it is consistent with the two edge examples due to $\{v_i,v_j\}$. Thus, we get

$$\begin{aligned} w_i + w_{n+i} + w_j + w_{n+j} &< 2t, \\ w_i + w_j + w_{n+i} + w_{n+j} &\geq 2t, \end{aligned}$$

a contradiction. This implies that V' is a vertex cover. ■

The reduction defined in the previous proof remains valid under the additional requirement that the threshold satisfies $t \leq 1$. As a consequence, we obtain the NP-completeness result claimed by Theorem 4.

Corollary 10 *For every $\ell \geq 1$,* MAX-$\mathcal{B}^\ell$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 2 *is NP-complete.*

Furthermore, we observe that the proof of Theorem 9 does not make use of the assumption that the weights must be from the set $\{0,1\}$. Consequently, the reduction works also for unconstrained weights, that is, for the function class $\mathcal{L}$. This establishes the NP-completeness statement of Theorem 5.

Corollary 11 MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE 1 AND SPARSENESS 2 *is NP-complete.*

5. Problems Solvable in Linear Time

In the following, we show that all consistency problems and maximum consistency problems not yet considered in the foregoing section have algorithms that run in linear time on a RAM as made precise in Section 2.4. Thus, if P $\neq$ NP holds, the values of coincidence and sparseness established in the NP-completeness results are optimal. Moreover, as any algorithm that solves a consistency or maximum consistency problem must pass through the entire sample, the linear-time bound cannot be improved either. Thus, we not only obtain a complete characterization of the complexity of consistency problems for the function classes $\mathcal{B}$, $\mathcal{B}^\ell$, and $\mathcal{L}$ with sample restrictions, but also reveal the striking fact that, with respect to the RAM computer model, these classes consist solely of problems of the lowest and highest complexity in NP. All algorithms presented here are constructive, that is, they solve the decision problem by computing a solution if one exists.

5.1 Consistency

It is convenient to begin with the consistency problem for neurons with a bounded threshold. The case that remains to be considered is the function class $\mathcal{B}^1$. The result for this class implies the second part of Theorem 2. It is valid for samples with arbitrary coincidence and sparseness.

Theorem 12 $\mathcal{B}^1$-CONSISTENCY *is solvable in linear time.*

Proof If there are weights $w_1,\ldots,w_n \in \{0,1\}$ and a threshold $t \in \{0,1\}$ consistent with a given sample S then we may set $t = 0$ if $\mathrm{Neg}(S) = \emptyset$. On the other hand, if S contains negative examples, we must have $t = 1$. Then we ensure that $w \cdot x = 0$ for all $x \in \mathrm{Neg}(S)$. These are the steps performed by Algorithm 1, which is basically the standard consistency algorithm for disjunctions. The most time consuming part is the for-loop in lines 8–12. It is obvious that this statement can be implemented such that it requires no more than linear time on a RAM. ■

It is easy to see that if some function in $\mathcal{B}$ is consistent with a sample S that has coincidence 0, the threshold can be chosen to satisfy $t \leq 1$. Therefore, Algorithm 1 solves also the problem $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0.

Corollary 13 $\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 *is solvable in linear time.*

Thus, the first of the two statements in Theorem 1 that claim a linear time bound is covered. The second statement is established by the following result.

Algorithm 1 $\mathcal{B}^1$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$
Output: $w_1, \ldots, w_n, t$

```
 1: for i = 1, ..., n do
 2:     w_i := 1
 3: end for;
 4: if Neg(S) = ∅ then
 5:     t := 0;
 6: else
 7:     t := 1;
 8:     for all x ∈ Neg(S) do
 9:         if there is some i ∈ {1, ..., n} with x_i = 1 then
10:             w_i := 0
11:         end if
12:     end for
13: end if.
```

Theorem 14 $\mathcal{B}$-CONSISTENCY WITH SPARSENESS 3 *is solvable in linear time.*

Proof We show that Algorithm 2 computes a solution for samples with sparseness 3 provided that one exists. Without loss of generality we may assume that the threshold satisfies $t \leq 3$. The algorithm has three parts that correspond to the possible values for the threshold: $t \leq 1$, $t = 3$, and $t = 2$. In lines 1–2 Algorithm 1 is used to check whether there is a function in the class $\mathcal{B}^1$ consistent with S.

If this fails, the algorithm assumes in lines 3–12 that $t = 3$. Here, the assignment of values to the weights is based on the fact that every positive example must reach the threshold. Hence, its non-zero components, of which there must be exactly three, specify which weights receive the value 1.

In the last part, starting from line 13, we can only have that $t = 2$. We show that this case of the consistency problem can be solved via a 2-SATISFIABILITY problem. The algorithm maps the sample S to a set $\mathcal{C}$ of clauses over the variables $w_1, \ldots, w_n$. Each clause has at most two literals of the form w_i or $\neg w_i$. For each element $x \in \mathrm{Dom}(S)$ the set $\mathcal{C}$ is augmented with clauses that depend on the properties of x. Lines 17–23 deal with the case that x has a 1 in exactly three positions. If x is a positive example then no two weights at these positions may both be equal to 0. This is expressed by the three clauses in line 20. If x is a negative example then no two of these weights may both be equal to 1. This is taken into account by adding the three clauses in line 22. Lines 24–30 treat the examples with exactly two 1s. If x is positive then both weights must be equal to 1 (line 27); if x is negative then at least one weight must be 0 (line 29). Finally, if x has at most one non-zero position and x is positive then the consistency problem has no solution. This is taken care of in line 34 by adding two contradictory clauses. On the other hand, if x is negative then it will be classified as such and nothing needs to be done. In the last step an algorithm for 2-SATISFIABILITY is called with the constructed set of clauses as input.

It remains to verify that the algorithm runs in linear time on a RAM. Using Theorem 12 we infer that the part in lines 1–14 requires no more than linear time. The construction of the set of clauses in lines 15–37 is performed in linear time. (Note that each example gives rise to at most three clauses.)

Algorithms that compute satisfying assignments for 2-SATISFIABILITY problems in linear time on a RAM with uniform measure have been designed by Even et al. (1976) and Aspvall et al. (1979). It is obvious that the size of the constructed instance for 2-SATISFIABILITY is at most linear in the size of the sample. Thus, line 38 can be executed in linear time. We conclude that the running time of Algorithm 2 on a random access machine is linear. ■

With the previous result the proof of Theorem 1 is completed. Obviously, Algorithm 2 solves also the consistency problems with sparseness 3 for the classes $\mathcal{B}^\ell$ where $\ell \geq 2$. (If $\ell = 2$, the middle part of the algorithm, which assumes $t = 3$, is needless.) Furthermore, if some function in $\mathcal{B}^\ell$ is consistent with a sample having coincidence 0 then there is also some function in $\mathcal{B}^1$ consistent with that sample. Consequently, Algorithm 1 solves also the problems $\mathcal{B}^\ell$-CONSISTENCY WITH COINCIDENCE 0 for every $\ell \geq 2$. Hence, Theorems and 14 and 12 yield the following statement. It finalizes the proof of Theorem 2.

Corollary 15 *For every $\ell \geq 2$, $\mathcal{B}^\ell$-*CONSISTENCY WITH COINCIDENCE 0 *and $\mathcal{B}^\ell$-*CONSISTENCY WITH SPARSENESS 3 *is solvable in linear time.*

5.2 Maximum Consistency

At last, we give attention to the maximum consistency problems that are claimed as linear-time solvable in Theorems 3, 4, and 5. The main issues that have to be dealt with are contradictory pairs and examples that are 0 in all positions. With the first result we finish the proof of Theorem 3.

Theorem 16 *The problems* MAX-$\mathcal{B}$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{B}$-CONSISTENCY WITH SPARSENESS 1 *are solvable in linear time.*

Proof If $\mathrm{Neg}(S) = \emptyset$ then by letting $w_1 = \cdots = w_n = 0$ and $t = 0$ we obtain the function that always outputs 1, which is consistent with all examples. Assume now that $\mathrm{Neg}(S) \neq \emptyset$. Algorithm 3 generates a function that has the following properties: For each example $x \in \mathrm{Dom}(S)$ that is part of a contradictory pair, that is, where $(x,0) \in S$ and $(x,1) \in S$, the function is consistent with the positive example. Further, it is consistent with all examples that are not part of a contradictory pair, except possibly for the example $(0,\ldots,0) \in \mathrm{Pos}(S) \setminus \mathrm{Neg}(S)$, which is classified as negative. It is obvious that this is the maximum number of examples that can be classified correctly, unless $(0,\ldots,0) \in \mathrm{Pos}(S) \setminus \mathrm{Neg}(S)$ is the only example that is not part of a contradictory pair. In this case we can increase the number of correctly classified examples by using the function that constantly outputs 1, which is then consistent with the positive example in each contradictory pair and with $(0,\ldots,0) \in \mathrm{Pos}(S)$.

It is obvious that Algorithm 3 and the additional steps described above require no more than linear time on a RAM. In particular, whether $(0,\ldots,0)$ is the only example that is not part of a contradictory pair can be checked in linear time using radix sort.

If $\mathrm{Sparse}(S) = 1$ then we proceed in the same way. In order to see that this is correct, observe that $\mathrm{Coin}(S) = 1$ holds only if S contains examples $(x,0)$ and $(x,1)$. Thus, a maximum subsample S' that does not contain contradictory pairs satisfies $\mathrm{Coin}(S') = 0$. ■

Algorithm 2 $\mathcal{B}$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$
Output: $w_1, \ldots, w_n, t$

get $w_1, \ldots, w_n, t$ from $\mathcal{B}^1$-CONSISTENCY(S);
if $(w_1, \ldots, w_n, t)$ is consistent with S **then stop end if**;
$t := 3$;
for $i = 1, \ldots, n$ **do**
  $w_i := 0$
end for;
for all $x \in \mathrm{Pos}(S)$ **do**
  if there is some $x \in \mathrm{Pos}(S)$ with $x_i = 1$ **then**
    $w_i := 1$
  end if
end for;
if $(w_1, \ldots, w_n, t)$ is consistent with S **then stop end if**;
$t := 2$;
$\mathcal{C} := \emptyset$;
for all $x \in \mathrm{Dom}(S)$ **do** {construct a set $\mathcal{C}$ of clauses over $w_1, \ldots, w_n$}
  let $I = \{i \mid x_i = 1\}$;
  if $|I| = 3$ **then**
    let $\{j,k,\ell\} = I$;
    if $x \in \mathrm{Pos}(S)$ **then**
      $\mathcal{C} := \mathcal{C} \cup \big\{\{w_j, w_k\}, \{w_j, w_\ell\}, \{w_k, w_\ell\}\big\}$
    else {we have $x \in \mathrm{Neg}(S)$}
      $\mathcal{C} := \mathcal{C} \cup \big\{\{\neg w_j, \neg w_k\}, \{\neg w_j, \neg w_\ell\}, \{\neg w_k, \neg w_\ell\}\big\}$
    end if
  else if $|I| = 2$ **then**
    let $\{j,k\} = I$;
    if $x \in \mathrm{Pos}(S)$ **then**
      $\mathcal{C} := \mathcal{C} \cup \big\{\{w_j\}, \{w_k\}\big\}$
    else {we have $x \in \mathrm{Neg}(S)$}
      $\mathcal{C} := \mathcal{C} \cup \big\{\{\neg w_j, \neg w_k\}\big\}$
    end if
  else {we have $|I| \leq 1$}
    if $|I| = 1$ **then** let $\{j\} = I$ **else** $j := 1$ **end if**;
    if $x \in \mathrm{Pos}(S)$ **then**
      $\mathcal{C} := \mathcal{C} \cup \big\{\{w_j\}, \{\neg w_j\}\big\}$
    end if
  end if
end for;
get $w_1, \ldots, w_n$ from 2-SATISFIABILITY($\{w_1, \ldots, w_n\}, \mathcal{C}$).

The algorithm presented in the foregoing proof generates only solutions where the threshold is at most 1. Thus, it solves also the corresponding problems for neurons with a bounded threshold as was claimed in the first statement of Theorem 4.

Algorithm 3 MAX-$\mathcal{B}$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$
Output: $w_1, \ldots, w_n, t$

```
t := 1;
for i = 1, ..., n do
   w_i := 0
end for;
for all x ∈ Pos(S) do
   if there is some i ∈ {1, ..., n} with x_i = 1 then
      w_i := 1
   end if
end for.
```

Corollary 17 *For every $\ell \geq 1$,* MAX-$\mathcal{B}^\ell$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{B}^\ell$-CONSISTENCY WITH SPARSENESS 1 *is solvable in linear time.*

The following result completes the proof of Theorem 4.

Lemma 18 MAX-$\mathcal{B}^0$-CONSISTENCY *is solvable in linear time.*

Proof If the threshold is equal to 0 then everything is classified as positive regardless of which values the weights may have. Thus, outputting any weights and $t = 0$ provides a solution if one exists. For solving the decision problem, observe that consistency with at least k examples in S is equivalent to the condition $|\mathrm{Pos}(S)| \geq k$. This can clearly be checked in linear time on a RAM. ■

Finally, we prove the remaining part of Theorem 5 that claims linear time bounds.

Theorem 19 *The problems* MAX-$\mathcal{L}$-CONSISTENCY WITH COINCIDENCE 0 *and* MAX-$\mathcal{L}$-CONSISTENCY WITH SPARSENESS 1 *are solvable in linear time.*

Proof Consider Algorithm 4 for inputs satisfying $\mathrm{Coin}(S) = 0$. If $\mathrm{Neg}(S) = \emptyset$ then the resulting function is consistent with all examples. In the case that $\mathrm{Neg}(S) \neq \emptyset$ holds we argue as in the proof of Theorem 16: For every contradictory pair, the function is consistent with the positive example. Further, it is consistent with every example that is not part of a contradictory pair. (In contrast to the proof of Theorem 16, this holds also in all cases where $(0, \ldots, 0) \in \mathrm{Dom}(S)$ due to the use of negative weights by Algorithm 4.) Clearly, this is the maximum number of examples a function can be consistent with.

Since any subsample S' of S that contains no contradictory pairs must satisfy $\mathrm{Coin}(S') = 0$ if $\mathrm{Sparse}(S) = 1$, Algorithm 4 solves also the maximum consistency problem for samples with sparseness 1.

That the running time of Algorithm 4 is linear on a RAM is obvious. ■

Algorithm 4 MAX-$\mathcal{L}$-CONSISTENCY

Input: sample $S \subseteq \{0,1\}^n \times \{0,1\}$
Output: $w_1, \ldots, w_n, t$
  if $(0, \ldots, 0) \in \mathrm{Pos}(S)$ **then**
    $t := 0$
  else
    $t := 1$
  end if;
  for $i = 1, \ldots, n$ **do**
    $w_i := -1$
  end for;
  for all $x \in \mathrm{Pos}(S)$ **do**
    if there is some $i \in \{1, \ldots, n\}$ with $x_i = 1$ **then**
      $w_i := 1$
    end if
  end for.

6. Conclusions

With this work we aimed at providing necessary and sufficient conditions for the existence of efficient learning algorithms. The approach that we have taken is to consider criteria for the complexity of samples and to use them for the definition of subproblems. As results we obtained dichotomy theorems that completely characterize the dividing lines between the polynomial-time solvable and the NP-complete problems.

An NP-completeness assertion is a worst-case result stating that, if P $\neq$ NP then there is no algorithm that solves the problem in polynomial time for all instances. One way to cope with intractability is therefore trying to avoid instances that make the problem hard. This gives rise to subproblems that may be more appropriate to practical situations where one often needs not deal with all possible instances but only a subset thereof. The results here show that such deliberations can indeed lead to the discovery of new efficient learning algorithms. While it is not clear whether the restrictions they suppose are always met in applications, these algorithms could be helpful in the development of better heuristics for more general learning problems.

A number of questions arises from this work that are worth further investigation. We have considered only two types of neural "networks": single neurons with binary and with arbitrary weights. Whether dichotomy theorems can be established for other neuron types and for neural networks is an interesting open problem. Further, the problems we have studied can be generalized by allowing examples with integer or even rational components. While the NP-completeness results remain valid in these cases, it is not obvious how the algorithms can be adapted to deal with non-binary inputs. Finally, we have introduced here two specific criteria for sample restrictions. Motivated by theoretical or practical considerations one can think of many other ways to select such conditions.

Acknowledgments

I thank the anonymous reviewers for their helpful comments and suggestions. 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. An NP-Complete Satisfiability Problem

Lemma 6 in Section 4.1 claims that the problem ALMOST DISJOINT POSITIVE 1-IN-3SAT introduced in that section is NP-complete. Here we give a proof of this statement.

Lemma 6 ALMOST DISJOINT POSITIVE 1-IN-3SAT *is NP-complete.*

Proof It is clear that the problem is in NP. To prove its NP-hardness, we construct a reduction from POSITIVE 1-IN-3SAT. In this variant of the problem the requirement is missing that two subsets have at most one variable in common. The idea of the reduction is to establish the property of almost disjointness by introducing new variables as substitutes for old ones. The correctness of the reduction is ensured by augmenting the collection with additional subsets.

Let $\mathcal{C}$ be the collection of subsets from the instance of POSITIVE 1-IN-3SAT. Further, let $C, D \in \mathcal{C}$ be two subsets with two common variables, say $C = \{u_1, u_2, u_3\}$ and $D = \{u_1, u_2, u_4\}$. We introduce a set of new variables $V = \{v_1, \ldots, v_5\}$, replace in D the variable u_1 by v_1 and join $\mathcal{C}$ with the collection $\mathcal{D}$ defined as

$$\mathcal{D} \;=\; \big\{\{v_1, v_2, v_3\},\ \{v_1, v_4, v_5\},\ \{u_1, v_2, v_4\},\ \{u_1, v_3, v_5\}\big\}.$$

Obviously, no pair of subsets in $\mathcal{D}$ has more than one common variable. Further, no subset in $\mathcal{D}$ has more than one variable in common with any subset in $\mathcal{C}$. Hence, going from $\mathcal{C}$ to $\mathcal{C} \cup \mathcal{D}$, the number of pairs that violate the condition of almost disjointness decreases by one.

We say that a truth assignment is a 1-IN-3SAT assignment for a collection of subsets if each subset has exactly one true variable. Let U be the set of variables in $\mathcal{C}$. We claim that the following two conditions hold:

(A.1) Every 1-IN-3SAT assignment $\alpha : U \to \{0,1\}$ for $\mathcal{C}$ can be extended to a 1-IN-3SAT assignment $\alpha : U \cup V \to \{0,1\}$ for $\mathcal{C} \cup \mathcal{D}$ satisfying $\alpha(u_1) = \alpha(v_1)$.

(A.2) Every 1-IN-3SAT assignment $\alpha : U \cup V \to \{0,1\}$ for $\mathcal{C} \cup \mathcal{D}$ satisfies $\alpha(u_1) = \alpha(v_1)$.

That condition (A.1) is valid can be seen as follows: Given α defined on U, we let $\alpha(v_1) = \alpha(u_1)$ and extend it to the remaining variables of V in the following way: In the case $\alpha(u_1) = 0$ we define $\alpha(v_2) = \alpha(v_5) = 0$ and $\alpha(v_3) = \alpha(v_4) = 1$; in the case $\alpha(u_1) = 1$ we let $\alpha(v_2) = \alpha(v_3) = \alpha(v_4) = \alpha(v_5) = 0$. For the proof of condition (A.2) we observe that if $\alpha(v_1) = 1$, we must have $\alpha(v_2) = \alpha(v_3) = \alpha(v_4) = \alpha(v_5) = 0$ due to the first two subsets in $\mathcal{D}$, and hence, because of the last two subsets, $\alpha(u_1) = 1$; on the other hand, if $\alpha(u_1) = 1$, analogous reasoning yields that $\alpha(v_1) = 1$ must hold. Thus, u_1 and v_1 cannot have different truth values assigned by α.

Conditions (A.1) and (A.2) imply that $\mathcal{C}$ has a 1-IN-3SAT assignment if and only if $\mathcal{C} \cup \mathcal{D}$ has a 1-IN-3SAT assignment. Repeating the above described procedure for every pair of subsets with two common variables we finally arrive at a collection that is almost disjoint. The correctness of

the reduction follows inductively. It is obvious that the reduction is computable in polynomial time. Thus, ALMOST DISJOINT POSITIVE 1-IN-3SAT is NP-hard. ■

References

Amaldi, E. (1991). On the complexity of training Perceptrons. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, pages 55–60. Elsevier, Amsterdam.

Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge.

Aspvall, B., Plass, M. F., and Tarjan, R. E. (1979). A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8:121–123.

Blum, A. L. and Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117–127.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36:929–965.

Cook, S. A. and Reckhow, R. A. (1973). Time bounded random access machines. *Journal of Computer and System Sciences*, 7:354–375.

Dalmau, V. (1999). A dichotomy theorem for learning quantified Boolean formulas. *Machine Learning*, 35:207–224.

Dalmau, V. and Jeavons, P. (2003). Learnability of quantified formulas. *Theoretical Computer Science*, 306:485–511.

Even, S., Itai, A., and Shamir, A. (1976). On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5:691–703.

Fang, S. C. and Venkatesh, S. S. (1996). Learning binary Perceptrons perfectly efficiently. *Journal of Computer and System Sciences*, 52:374–389.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco.

Golea, M. and Marchand, M. (1993). On learning Perceptrons with binary weights. *Neural Computation*, 5:767–782.

Hampson, S. E. and Volper, D. J. (1986). Linear function neurons: Structure and training. *Biological Cybernetics*, 53:203–217.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, second edition.

Höffgen, K.-U., Simon, H.-U., and Van Horn, K. S. (1995). Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50:114–125.

Judd, J. S. (1990). *Neural Network Design and the Complexity of Learning*. The MIT Press, Cambridge, MA.

Kearns, M. J., Schapire, R. E., and Sellie, L. M. (1994). Toward efficient agnostic learning. *Machine Learning*, 17:117–141.

Kim, J. H. and Roche, J. R. (1998). Covering cubes by random half cubes, with applications to binary neural networks. *Journal of Computer and System Sciences*, 56:223–252.

Kirousis, L. M. and Kolaitis, P. G. (2003). The complexity of minimal satisfiability problems. *Information and Computation*, 187:20–39.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

Maass, W. and Turán, G. (1994). How fast can a threshold gate learn? In Hanson, S. J., Drastal, G., and Rivest, R., editors, *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects*, pages 381–414. The MIT Press, Cambridge, MA.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Natarajan, B. K. (1991). *Machine Learning: A Theoretical Approach*. Morgan Kaufmann, San Mateo, CA.

Palm, G. (1980). On associative memory. *Biological Cybernetics*, 36:19–31.

Pitt, L. and Valiant, L. G. (1988). Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35:965–984.

Schaefer, T. J. (1978). The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226.

Schmitt, M. (1994a). On the complexity of consistency problems for neurons with binary weights. Ulmer Informatik-Berichte 94-01, Universität Ulm.

Schmitt, M. (1994b). On the size of weights for McCulloch-Pitts neurons. In Caianiello, E. R., editor, *Proceedings of the Sixth Italian Workshop on Neural Nets WIRN VIETRI-93*, pages 241–246, World Scientific, Singapore.

Schmitt, M. (1995). On methods to keep learning away from intractability. In Fogelman-Soulié, F. and Gallinari, P., editors, *Proceedings of the International Conference on Artificial Neural Networks ICANN '95*, volume 1, pages 211–216, EC2 & Cie., Paris.

Servedio, R. A. (2002). Perceptron, Winnow, and PAC learning. *SIAM Journal on Computing*, 31:1358–1369.

Sommer, F. T. and Palm, G. (1999). Improved bidirectional retrieval of sparse patterns stored by Hebbian learning. *Neural Networks*, 12:281–297.

van Emde Boas, P. (1990). Machine models and simulations. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, chapter 1, pages 1–66. Elsevier, Amsterdam.

Šíma, J. (2002). Training a single sigmoidal neuron is hard. *Neural Computation*, 14:2709–2728.

Zheng, L. and Stuckey, P. J. (2002). Improving SAT using 2SAT. In Oudshoorn, M. J., editor, *Proceedings of the Twenty-Fifth Australasian Computer Science Conference ACSC2002*, volume 4 of *Conferences in Research and Practice in Information Technology*, pages 331–340, Australian Computer Society, Sydney.

Journal of Machine Learning Research 5 (2004) 913–939 Submitted 7/03; Revised 11/03; Pulished 8/04

Image Categorization by Learning and Reasoning with Regions

Yixin Chen YIXIN@CS.UNO.EDU
Department of Computer Science
University of New Orleans
New Orleans, LA 70148, USA

James Z. Wang JWANG@IST.PSU.EDU
School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA

Editor: Donald Geman

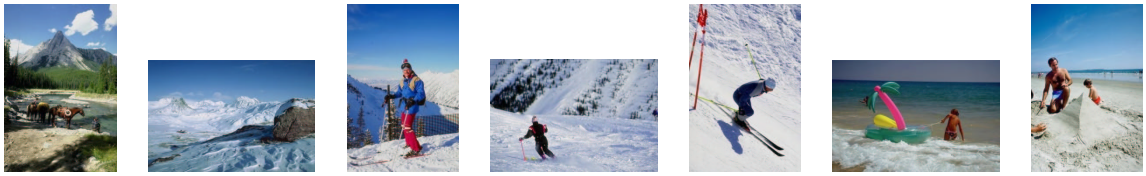
Abstract

Designing computer programs to automatically categorize images using low-level features is a challenging research topic in computer vision. In this paper, we present a new learning technique, which extends Multiple-Instance Learning (MIL), and its application to the problem of region-based image categorization. Images are viewed as bags, each of which contains a number of instances corresponding to regions obtained from image segmentation. The standard MIL problem assumes that a bag is labeled positive if at least one of its instances is positive; otherwise, the bag is negative. In the proposed MIL framework, DD-SVM, a bag label is determined by some number of instances satisfying various properties. DD-SVM first learns a collection of *instance prototypes* according to a Diverse Density (DD) function. Each instance prototype represents a class of instances that is more likely to appear in bags with the specific label than in the other bags. A nonlinear mapping is then defined using the instance prototypes and maps every bag to a point in a new feature space, named the *bag feature space*. Finally, standard support vector machines are trained in the bag feature space. We provide experimental results on an image categorization problem and a drug activity prediction problem.

Keywords: image categorization, multiple-instance learning, support vector machines, image classification, image segmentation

1. Introduction

The term image categorization refers to the labeling of images into one of a number of predefined categories. Although this is usually not a very difficult task for humans, it has proved to be an extremely difficult problem for machines (or computer programs). Major sources of difficulties include variable and sometimes uncontrolled imaging conditions, complex and hard-to-describe objects in an image, objects occluding other objects, and the gap between arrays of numbers representing physical images and conceptual information perceived by humans. In this paper, an object in the physical world, which we live in, refers to anything that is visible or tangible and is relatively stable in form. An object in an image is defined as a region, not necessarily connected, which is a projection of an object in the physical world. Designing automatic image categorization algorithms has been an important research field for decades. Potential applications include digital



(a) (b) (c) (d) (e) (f) (g)

Figure 1: Sample images belonging to one of the categories: Mountains and glaciers, Skiing, and Beach.

libraries, Space science, Web searching, geographic information systems, biomedicine, surveillance and sensor systems, commerce, and education.

1.1 Overview of Our Approach

Although color and texture are fundamental aspects for visual perception, human discernment of certain visual contents could be potentially associated with interesting classes of objects or semantic meaning of objects in the image. For one example: if we are asked to decide which images in Figure 1 are images about *Mountains and glaciers*, *Skiing*, and *Beach*, at a single glance, we may come up with the following answers together with supporting arguments:

- Images (a) and (b) are images about mountains and glaciers since we see *mountain* in them;
- Images (c), (d) and (e) are skiing images since there are *snow*, *people*, and perhaps a steep *slope* or *mountain* in them;
- Images (f) and (g) are beach images since we see either *people* playing in *water* or *people* on *sand*;

This seems to be effortless for humans because prior knowledge of similar images and objects may provide powerful assistance for us in recognition. Given a set of labeled images, can a computer program learn such knowledge or semantic concepts from implicit information of objects contained in images? This is the question we attempt to address in this work.

In terms of image representation, our approach is a region-based method. Images are segmented into regions such that each region is roughly homogeneous in color and texture. Each region is characterized by one feature vector describing color, texture, and shape attributes. Consequently, an image is represented by a collection of feature vectors. If segmentation is ideal, regions will correspond to objects. But, in general, semantically accurate image segmentation by a computer program is still an ambitious long-term goal for computer vision researchers (see Shi and Malik, 2000; Wang et al., 2001a; Zhu and Yuille, 1996). Here, semantically accurate image segmentation refers to building a one-to-one mapping between regions generated by an image segmentation algorithm and objects in the image. Nevertheless, we argue that region-based image representation can provide some useful information about objects even though segmentation may not be perfect. Moreover, empirical results in Section 4.3 demonstrate that the proposed method has low sensitivity to inaccurate image segmentation.

From the perspective of learning, our approach is a generalization of supervised learning, in which labels are associated with images instead of individual regions. This is in essence identical to

the Multiple-Instance Learning (MIL) setting (Dietterich et al., 1997; Blum and Kalai, 1998; Maron and Lozano-Pérez, 1998; Zhang and Goldman, 2002) where images and regions are respectively called *bags* and *instances*. In this paper, a "bag" refers to an "image", and an "instance" refers to a "region." MIL assumes that every instance possesses an unknown label that is indirectly accessible through labels attached to bags.

1.2 Related Work in Multiple-Instance Learning

Several researchers have applied MIL for image classification and retrieval (Andrews et al., 2003; Maron and Ratan, 1998; Zhang et al., 2002; Yang and Lozano-Pérez, 2000). Key assumptions of their formulation of MIL are that bags and instances share the same set of labels (or categories or classes or topics), and a bag receives a particular label if at least one of the instances in the bag possesses the label. For binary classification, this implies that a bag is "positive" if at least one of its instances is a positive example; otherwise, the bag is "negative." Therefore, learning focuses on finding "actual" positive instances in positive bags. The formulations of MIL in image classification and retrieval fall into two categories: the Diverse Density approach (Maron and Ratan, 1998; Zhang et al., 2002) and the Support Vector Machine (SVM) approach (Andrews et al., 2003).

- In the Diverse Density approach, an objective function, called the Diverse Density (DD) function (Maron and Lozano-Pérez, 1998), is defined over the instance feature space, in which instances can be viewed as points. The DD function measures a co-occurrence of similar instances from different bags with the same label. A feature point with large Diverse Density indicates that it is close to at least one instance from every positive bag and far away from every negative instance. The DD approach searches the instance feature space for points with high Diverse Density. Once a point with the maximum DD is found, a new bag is classified according to the distances between instances in the bag and the maximum DD point: if the smallest distance is less than certain fixed threshold, the bag is classified as positive; otherwise, the bag is classified as negative. The major difference between Maron's method and Zhang's method lies in the way to search a maximum DD point. Zhang's method is relatively insensitive to the dimension of instance feature space and scales up well to the average bag size, i.e., the average number of instances in a bag (Zhang and Goldman, 2002). Empirical studies demonstrate that DD-based MIL can learn certain simple concepts of natural scenes, such as *waterfall*, *sunsets*, and *mountains*, using features of subimages or regions (Maron and Ratan, 1998; Zhang et al., 2002).

- Andrews et al. (2003) use SVMs (Burges, 1998) to solve the MIL problem. In particular, MIL is formulated as a mixed integer quadratic program. In their formulation, integer variables are selector variables that select which instance in a positive bag is the positive instance. Their algorithm, which is called MI-SVM, has an outer loop and an inner loop. The outer loop sets the values of these selector variables. The inner loop then trains a standard SVM in which the selected positive instances replace the positive bags. The outer loop stops if none of the selector variables changes value in two consecutive iterations. Andrews et al. (2003) show that MI-SVM outperforms the DD approach described in Zhang and Goldman (2002) on a set of images belonging to three different categories ("elephant", "fox", and "tiger"). The difference between MI-SVM and DD approach can also be viewed from the shape of the corresponding classifier's decision boundary in the instance feature space. The decision boundary of a DD

classifier is an ellipsoidal sphere because classification is based exclusively on the distance to the maximum DD point.[1] For MI-SVM, depending on the kernel used, the decision boundary can be a hyperplane in the instance feature space or a hyperplane in the kernel induced feature space, which may correspond to very complex boundaries in the instance feature space.

1.3 A New Formulation of Multiple-Instance Learning

In the above MIL formulations, a bag is essentially summarized by one of its instances, i.e., an instance with the maximal label (considering binary classification with 1 and -1 representing the positive and negative classes, respectively). However, these formulations have a drawback for image categorization tasks: in general, a concept about images may not be captured by a single region (instance) even if image segmentation and object recognition are assumed to be ideal (inaccurate segmentation and recognition will only worsen the situation). For one simple example, let's consider categorizing *Mountains and glaciers* versus *Skiing* images in Figure 1. To classify a scene as involving skiing, it is helpful to identify *snow*, *people*, and perhaps *mountain*. If an image is viewed as a bag of regions, then the standard MIL formulation cannot realize this, because a bag is labeled positive if any one region in the bag is positive. In addition, a class might also be disjunctive. As shown by Figure 1 (g) and (h), a *Beach* scene might involve either *people* playing in *water* or *people* on *sand*. Thus we argue that the correct categorization of an image depends on identifying multiple aspects of the image. This motivates our extension of MIL where a bag must contain a number of instances satisfying various properties (e.g., *people*, *snow*, etc.).

In our approach, MIL is formulated as a maximum margin problem in a new feature space defined by the DD function. The new approach, named DD-SVM, proceeds in two steps. First, in the instance feature space, a collection of feature vectors, each of which is called an *instance prototype*, is determined according to DD. Each instance prototype is chosen to be a local maximizer of the DD function. Since DD measures the co-occurrence of similar instances from different bags with the same label, loosely speaking, an instance prototype represents a class of instances (or regions) that is more likely to appear in bags (or images) with the specific label than in the other bags (or images). Second, a nonlinear mapping is defined using the learned instance prototypes and maps every bag to a point in a new feature space, which is named the *bag feature space*. In the bag feature space, the original MIL problem becomes an ordinary supervised learning problem. Standard SVMs are then trained in the bag feature space.

DD-SVM is similar to MI-SVM in the sense that both approaches apply SVM to solve the MIL problem. However, in DD-SVM, several features are defined for each bag. Each bag feature could be defined by a separate instance within the bag (i.e., the instance that is most similar to an instance prototype). Hence, the bag features summarize the bag along several dimensions defined by instance prototypes. This is in stark contrast to MI-SVM, in which one instance is selected to represent the whole positive bag.

1.4 Related Work in Image Categorization

In the areas of image processing, computer vision, and pattern recognition, there has been abundance of prior work on detecting, recognizing, and classifying a relatively small set of objects or concepts

1. The maximum DD algorithms described in (Maron and Lozano-Pérez, 1998; Zhang and Goldman, 2002) produce a point in the instance feature space together with scaling factors for each feature dimension. Therefore, the decision boundary is an ellipsoidal sphere instead of a sphere.

in specific domains of application (Forsyth and Ponce, 2002; Marr, 1983; Strat, 1992). We only review work most relevant to what we propose, which by no means represents the comprehensive list in the cited area.

As one of the simplest representations of digital images, histograms have been widely used for various image categorization problems. Szummer and Picard (1998) use *k*-nearest neighbor classifier on color histograms to discriminate between *indoor* and *outdoor* images. In the work of Vailaya et al. (2001), Bayesian classifiers using color histograms and edge directions histograms are implemented to organize *sunset/forest/mountain* images and *city/landscape* images, respectively. Chapelle et al. (1999) apply SVMs, which are built on color histogram features, to classify images containing a generic set of objects. Although histograms can usually be computed with little cost and are effective for certain classification tasks, an important drawback of a global histogram representation is that information about spatial configuration is ignored. Many approaches have been proposed to tackle the drawback. In the method of Huang et al. (1998), a classification tree is constructed using color correlograms. Color correlogram captures the spatial correlation of colors in an image. Gdalyahu and Weinshall (1999) apply local curve matching for shape silhouette classifications, in which objects in images are represented by their outlines.

A number of subimage-based methods have been proposed to exploit local and spatial properties by dividing an image into rectangular blocks. In the method introduced by Gorkani and Picard (1994), an image is first divided into 16 non-overlapping equal-sized blocks. Dominant orientations are computed for each block. The image is then classified as *city* or *suburb* scenes as determined by the majority orientations of blocks. Wang et al. (2001b) develop a *graph/photograph* classification algorithm.[2] The classifier partitions an image into blocks and classifies every block into one of two categories based on wavelet coefficients in high frequency bands. If the percentage of blocks classified as photograph is higher than a threshold, the image is marked as a photograph; otherwise, the image is marked as a graph. Yu and Wolf (1995) present a one-dimensional Hidden Markov Model (HMM) for *indoor/outdoor* scene classification. The model is trained on vector quantized color histograms of image blocks. In the recent ALIP system (Li and Wang, 2003), a concept corresponding to a particular category of images is captured by a two-dimensional multiresolution HMM trained on color and texture features of image blocks. Murphy et al. (2004) propose four graphical models that relate features of image blocks to objects and perform joint scene and object recognition.

Although a rigid partition of an image into rectangular blocks preserves certain spatial information, it often breaks an object into several blocks or puts different objects into a single block. Thus visual information about objects, which could be beneficial to image categorization, may be destroyed by a rigid partition. The ALIP system (Li and Wang, 2003) uses a small block size (4×4) for feature extraction to avoid this problem. Image segmentation is one way to extract object information. It decomposes an image into a collection of regions, which correspond to objects if decomposition is ideal. Segmentation-based algorithms can take into consideration the shape information, which is in general not available without segmentation.

Image segmentation has been successfully used in content-based image and video analysis (e.g., Carson et al., 2002; Chen and Wang, 2002; Ma and Manjunath, 1997; Modestino and Zhang, 1992; Smith and Li, 1999; Vasconcelos and Lippman, 1998; Wang et al., 2001b). Modestino and Zhang (1992) apply a Markov random field model to capture spatial relationships between regions. Im-

2. As defined by Wang et al. (2001b), a graph image is an image containing mainly text, graph, and overlays; a photograph is a continuous-tone image.

age interpretation is then given by a maximum *a posteriori* rule. SIMPLIcity system (Wang et al., 2001b) classifies images into *textured* or *nontextured* classes based upon how evenly a region scatters in an image. Mathematically, this is described by the goodness of match, which is measured by the χ^2 statistics, between the distribution of the region and a uniform distribution. Smith and Li (1999) propose a method for classifying images by spatial orderings of regions. Their system decomposes an image into regions with the attribute of interest of each region represented by a symbol that corresponds to an entry in a finite pattern library. The region string is converted to composite region template descriptor matrix that enables classification using spatial information. Vasconcelos and Lippman (1998) model image retrieval as a classification problem based on the principle of Bayesian inference. The information of the regions identified as human skin is used in the inference. Very interesting results have been achieved in associating words to images based on regions (Barnard and Forsyth, 2001) or relating words to image regions (Barnard et al., 2003). In their method, an image is modeled as a sequence of regions and a sequence of words generated by a hierarchical statistic model. The method demonstrates the potential for searching images. But as noted by Barnard and Forsyth (2001), the method relies on semantically meaningful segmentation, which, as mentioned earlier, is still an open problem in computer vision.

1.5 Outline of the Paper

The remainder of the paper is organized as follows. Section 2 describes image segmentation and feature representation. Section 3 presents DD-SVM, an extension of MIL. Section 4 describes the extensive experiments we have performed and provides the results. Finally, we conclude in Section 5, together with a discussion of future work.

2. Image Segmentation and Representation

In this section we describe a simple image segmentation procedure based on color and spatial variation features using a *k*-means algorithm (Hartigan and Wong, 1979). For general-purpose images such as the images in a photo library or images on the World Wide Web, precise object segmentation is nearly as difficult as natural language semantics understanding. However, semantically precise segmentation is not crucial to our system. As we will demonstrate in Section 4, our image categorization method has low sensitivity to inaccurate segmentation. Image segmentation is a well-studied topic (e.g., Shi and Malik, 2000; Wang et al., 2001a; Zhu and Yuille, 1996). The focus of this paper is not to achieve superior segmentation results but good categorization performance. The major advantage of the proposed image segmentation is its low computational cost.

To segment an image, the system first partitions the image into non-overlapping blocks of size 4×4 pixels. A feature vector is then extracted for each block. The block size is chosen to compromise between texture effectiveness and computation time. Smaller block size may preserve more texture details but increase the computation time as well. Conversely, increasing the block size can reduce the computation time but lose texture information and increase the segmentation coarseness. Each feature vector consists of six features. Three of them are the average color components in a block. We use the well-known LUV color space, where L encodes luminance and U and V encode color information (chrominance). The other three represent square root of energy in the high-frequency bands of the wavelet transforms (Daubechies, 1992), that is, the square root of the second order moment of wavelet coefficients in high-frequency bands.

2 regions 3 regions 7 regions 2 regions 3 regions 4 regions

Figure 2: Segmentation results by the *k*-means clustering algorithm. First row: original images. Second row: regions in their representative colors.

To obtain these moments, a Daubechies-4 wavelet transform is applied to the L component of the image. After a one-level wavelet transform, a 4×4 block is decomposed into four frequency bands: the LL, LH, HL, and HH bands. Each band contains 2×2 coefficients. Without loss of generality, we suppose the coefficients in the HL band are $\{c_{k,l}, c_{k,l+1}, c_{k+1,l}, c_{k+1,l+1}\}$. One feature is

$$f = \left(\frac{1}{4} \sum_{i=0}^{1} \sum_{j=1}^{1} c_{k+i,l+j}^2 \right)^{\frac{1}{2}}.$$

The other two features are computed similarly to the LH and HH bands. Unser (1995) shows that moments of wavelet coefficients in various frequency bands are effective for representing texture. For example, the HL band shows activities in the horizontal direction. An image with vertical strips thus has high energy in the HL band and low energy in the LH band.

The *k*-means algorithm is used to cluster the feature vectors into several classes with every class corresponding to one "region" in the segmented image. No information about the spatial layout of the image is used in defining the regions, so they are not necessarily spatially contiguous. The algorithm does not specify the number of clusters, N, to choose. We adaptively select N by gradually increasing N until a stopping criterion is met. The number of clusters in an image changes in accordance with the adjustment of the stopping criteria. A detailed description of the stopping criteria can be found in Wang et al. (2001b). Examples of segmentation results are shown in Figure 2. Segmented regions are shown in their representative colors. It takes less than one second on average to segment a 384×256 image on a Pentium III 700MHz PC running the Linux operating system. Since it is almost impossible to find a stopping criterion that is best suited for a large collection of images, images sometimes may be under-segmented or over-segmented. However, our categorization method has low sensitivity to inaccurate segmentation.

After segmentation, the mean of the set of feature vectors corresponding to each region $\mathbf{R}_j$ (a subset of $\mathbb{Z}^2$) is computed and denoted as $\hat{\mathbf{f}}_j$. Three extra features are also calculated for each region to describe shape properties. They are normalized inertia (Gersho, 1979) of order 1, 2, and 3. For a

region $\mathbf{R}_j$ in the image plane, the normalized inertia of order γ is given as

$$I(\mathbf{R}_j,\gamma)=\frac{\sum_{\mathbf{r}\in\mathbf{R_j}}\|\mathbf{r}-\hat{\mathbf{r}}\|^{\gamma}}{V_j^{1+\frac{\gamma}{2}}},$$

where $\hat{\mathbf{r}}$ is the centroid of $\mathbf{R}_j$, V_j is the number of pixels in region $\mathbf{R}_j$. The normalized inertia is invariant to scaling and rotation. The minimum normalized inertia on a 2-dimensional plane is achieved by circles. Denote the γ-th order normalized inertia of circles as I_γ. We define shape features of region $\mathbf{R}_j$ as

$$\mathbf{s}_j=\left[\frac{I(\mathbf{R}_j,1)}{I_1},\frac{I(\mathbf{R}_j,2)}{I_2},\frac{I(\mathbf{R}_j,3)}{I_3}\right]^T.$$

Finally, an image $\mathbf{B}_i$, which is segmented into N_i regions $\{\mathbf{R}_j : j=1,\cdots,N_i\}$, is represented by a collection of feature vectors $\{\mathbf{x}_{ij} : j=1,\cdots,N_i\}$. Each $\mathbf{x}_{ij}$ is a 9-dimensional feature vector, corresponding to region $\mathbf{R}_j$, defined as

$$\mathbf{x}_{ij}=\left[\hat{\mathbf{f}}_j^T,\mathbf{s}_j^T\right]^T.$$

3. An Extension of Multiple-Instance Learning

In this section, we first introduce DD-SVM, a maximum margin formulation of MIL in bag feature space. We then describe one way to construct a bag feature space using Diverse Density. Finally, we compare DD-SVM with another SVM-based MIL formulation, MI-SVM, proposed by Andrews et al. (2003).

3.1 Maximum Margin Formulation of MIL in a Bag Feature Space

We start with some notations in MIL. Let $\mathcal{D}$ be the labeled data set, which consists of l bag/label pairs, i.e., $\mathcal{D}=\{(\mathbf{B}_1,y_1),\cdots,(\mathbf{B}_l,y_l)\}$. Each bag $\mathbf{B}_i\subset\mathbb{R}^m$ is a collection of instances with $\mathbf{x}_{ij}\in\mathbb{R}^m$ denoting the j-th instance in the bag. Different bags may have different number of instances. Labels y_i take binary values 1 or -1. A bag is called a positive bag if its label is 1; otherwise, it is called a negative bag. Note that a label is attached to each bag and not to every instance. In the context of images, a bag is a collection of region feature vectors; an instance is a region feature vector; positive (negative) label represents that an image belongs (does not belong) to a particular category.

The basic idea of the new MIL framework is to map every bag to a point in a new feature space, named the bag feature space, and to train SVMs in the bag feature space. For an introduction to SVMs, we refer interested readers to tutorials and books on this topic (Burges, 1998; Cristianini and Shawe-Taylor, 2000). The maximum margin formulation of MIL in a bag feature space is given as the following quadratic optimization problem:

$$DD-SVM \qquad \alpha^*=\arg\max_{\alpha_i}\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}y_iy_j\alpha_i\alpha_jK(\phi(\mathbf{B}_i),\phi(\mathbf{B}_j)) \tag{1}$$

$$subject\ to \quad \sum_{i=1}^{l}y_i\alpha_i=0$$

$$C\geq\alpha_i\geq 0,\ i=1,\cdots,l.$$

The bag feature space is defined by $\phi : \mathcal{B} \to \mathbb{R}^n$ where $\mathcal{B}$ is a subset of $\mathcal{P}(\mathbb{R}^m)$ (the power set of $\mathbb{R}^m$). In practice, we can assume that the elements of $\mathcal{B}$ are finite sets since the number of instances in a bag is finite. $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function. The parameter C controls the trade-off between accuracy and regularization. The bag classifier is then defined by α^* as

$$\text{label}(\mathbf{B}) = \text{sign}\left(\sum_{i=1}^{l} y_i \alpha_i^* K(\phi(\mathbf{B}_i), \phi(\mathbf{B})) + b^*\right) \tag{2}$$

where b^* is chosen so that

$$y_j\left(\sum_{i=1}^{l} y_i \alpha_i^* K(\phi(\mathbf{B}_i), \phi(\mathbf{B}_j)) + b^*\right) = 1$$

for any j with $C > \alpha_j^* > 0$. The optimization problem (1) assumes that the bag feature space (i.e., ϕ) is given. Next, we introduce a way of constructing ϕ from a set of labeled bags.

3.2 Constructing a Bag Feature Space

Given a set of labeled bags, finding what is in common among the positive bags and does not appear in the negative bags may provide inductive clues for classifier design. In our approach, such clues are captured by instance prototypes computed from the DD function. A bag feature space is then constructed using the instance prototypes, each of which defines one dimension of the bag feature space.

3.2.1 DIVERSE DENSITY

In the ideal scenario, the intersection of the positive bags minus the union of the negative bags gives the instances that appear in all the positive bags but none of the negative bags. However, in practice strict set operations of intersection, union, and difference may not be useful because most real world problems involve noisy information. Features of instances might be corrupted by noise. Some bags might be mistakenly labeled. Strict intersection of positive bags might generate the empty set. Diverse Density implements soft versions of the intersection, union, and difference operations by thinking of the instances and bags as generated by some probability distribution. It is a function defined over the instance feature space. The DD value at a point in the feature space is indicative of the probability that the point agrees with the underlying distribution of positive and negative bags.

Next, we introduce one definition of DD from Maron and Lozano-Pérez (1998). Interested readers are referred to Maron and Lozano-Pérez (1998) for detailed derivations based on a probabilistic framework. Given a labeled data set $\mathcal{D}$, the DD function is defined as

$$DD_{\mathcal{D}}(\mathbf{x}, \mathbf{w}) = \prod_{i=1}^{l}\left[\frac{1+y_i}{2} - y_i \prod_{j=1}^{N_i}\left(1 - e^{-\|\mathbf{x}_{ij}-\mathbf{x}\|_{\mathbf{w}}^2}\right)\right]. \tag{3}$$

Here, $\mathbf{x}$ is a point in the instance feature space; $\mathbf{w}$ is a weight vector defining which features are considered important and which are considered unimportant; N_i is the number of instances in the i-th bag; and $\|\cdot\|_{\mathbf{w}}$ denotes a weighted norm defined by

$$\|\mathbf{x}\|_{\mathbf{w}} = \left[\mathbf{x}^T \text{Diag}(\mathbf{w})^2 \mathbf{x}\right]^{\frac{1}{2}} \tag{4}$$

where Diag($\mathbf{w}$) is a diagonal matrix whose (i,i)-th entry is the i-th component of $\mathbf{w}$.

It is not difficult to observe that values of DD are always between 0 and 1. For fixed weights $\mathbf{w}$, if a point $\mathbf{x}$ is close to an instance from a positive bag $\mathbf{B}_i$, then

$$\frac{1+y_i}{2} - y_i \prod_{j=1}^{N_i} \left(1 - e^{-\|\mathbf{x}_{ij}-\mathbf{x}\|^2_{\mathbf{w}}}\right) \tag{5}$$

will be close to 1; if $\mathbf{x}$ is close to an instance from a negative bag $\mathbf{B}_i$, then (5) will be close to 0. The above definition indicates that $DD(\mathbf{x},\mathbf{w})$ will be close to 1 if $\mathbf{x}$ is close to instances from different positive bags and, at the same time, far away from instances in all negative bags. Thus it measures a co-occurrence of instances from different (diverse) positive bags.

3.2.2 LEARNING INSTANCE PROTOTYPES

The DD function defined in (3) is a continuous and highly nonlinear function with multiple peaks and valleys (or local maximums and minimums). A larger value of DD at a point indicates a higher probability that the point fits better with the instances from positive bags than with those from negative bags. This motivates us to choose local maximizers of DD as instance prototypes. Loosely speaking, an instance prototype represents a class of instances that is more likely to appear in positive bags than in negative bags. Note that, the MIL formulation in Maron and Lozano-Pérez (1998) computes the global maximizer of DD, which corresponds to one instance prototype in our notation.

Learning instance prototypes then becomes an optimization problem: finding local maximizers of the DD function in a high-dimensional space. For our application the dimension of the optimization problem is 18 because the dimension of the region features is 9 and the dimension of weights is also 9. Since the DD functions are smooth, we apply gradient based methods to find local maximizers. Now the question is: how do we find all the local maximizers? In general, we do not know the number of local maximizers a DD function has. However, according to the definition of DD, a local maximizer is close to instances from positive bags (Maron and Lozano-Pérez, 1998). Thus starting a gradient based optimization from one of those instances will likely lead to a local maximum. Therefore, a simple heuristic is applied to search for multiple maximizers: we start an optimization at every instance in every positive bag with uniform weights, and record all the resulting distinct maximizers (feature vector and corresponding weights).

Instance prototypes are selected from those maximizers with two additional constraints: (a) they need to be distinct from each other; and (b) they need to have large DD values. The first constraint addresses the precision issue of numerical optimization. Due to numerical precision, different starting points may lead to different versions of the same maximizer. Hence we need to remove some of the maximizers that are essentially repetitions of each other. The second constraint limits instance prototypes to those that are most informative in terms of co-occurrence in different positive bags. In our algorithm, this is achieved by picking maximizers with DD values greater than certain threshold.

Following the above descriptions, one can find instance prototypes representing classes of instances that are more likely to appear in positive bags than in negative bags. One could argue that instance prototypes with the exactly reversed property (more likely to appear in negative bags than in positive bags) may be of equal importance. Such instance prototypes can be computed in exactly the same fashion after negating the labels of positive and negative bags. Our empirical study shows that including such instance prototypes (for negative bags) improves classification accuracy by an average amount of 2.2% for the 10-class image categorization experiment described in Section 4.2.

3.2.3 AN ALGORITHMIC VIEW

Next, we summarize the above discussion in pseudo code. The input is a set of labeled bags $\mathcal{D}$. The following pseudo code learns a collection of instance prototypes each of which is represented as a pair of vectors $(\mathbf{x}_i^*, \mathbf{w}_i^*)$. The optimization problem involved is solved by Quasi-Newton search `dfpmin` in Press et al. (1992).

Algorithm 3.1 Learning Instance Prototypes

MainLearnIPs($\mathcal{D}$)

```
 1 I_p = LearnIPs(D)                    [Learn Instance Prototypes for positive bags]
 2 negate labels of all bags in D
 3 I_n = LearnIPs(D)                    [Learn Instance Prototypes for negative bags]
 4 OUTPUT (the set union of I_p and I_n)
```

LearnIPs($\mathcal{D}$)

```
 1 set P be the set of instances from all positive bags in D
 2 initialize M to be the empty set
 3 FOR (every instance in P as starting point for x)
 4    set the starting point for w to be all 1's
 5    find a maximizer (p,q) of the log(DD) function by quasi-Newton search
 6    add (p,q) to M
 7 END
 8 set i = 1, T = (max_{(p,q)∈M} log(DD_D(p,q)) + min_{(p,q)∈M} log(DD_D(p,q)))/2
 9 REPEAT
10    set (x_i*, w_i*) = arg max_{(p,q)∈M} log(DD_D(p,q))
11    remove from M all elements (p,q) satisfying
      ||p ⊗ abs(q) − x_i* ⊗ abs(w_i*)|| < β||x_i* ⊗ abs(w_i*)|| OR log(DD_D(p,q)) < T
12    set i = i + 1
13 WHILE (M is not empty)
14 OUTPUT ({(x_1*, w_1*), ..., (x_{i−1}*, w_{i−1}*)})
```

In the above pseudo code for **LearnIPs**, lines 1–7 find a collection of local maximizers for the DD function by starting optimization at every instance in every positive bag with uniform weights. For better numerical stability, the optimization is performed on the $\log(DD)$ function, instead of the DD function itself. In line 5, we implement the EM-DD algorithm (Zhang and Goldman, 2002), which scales up well to large bag sizes in running time. Lines 8–13 describe an iterative process to pick a collection of "distinct" local maximizers as instance prototypes. In each iteration, an element of $\mathbf{M}$, which is a local maximizer, with the maximal $\log(DD)$ value (or, equivalently, the DD value) is selected as an instance prototype (line 10). Then elements of $\mathbf{M}$ that are close to the selected instance prototype or that have DD values lower than a threshold are removed from $\mathbf{M}$ (line 11). A new iteration starts if $\mathbf{M}$ is not empty. The $\text{abs}(\mathbf{w})$ in line 11 computes component-wise absolute values of $\mathbf{w}$. This is because the signs in $\mathbf{w}$ have no effect on the definition (4) of weighted norm. The $\otimes$ in line 11 denotes component-wise product.

The number of instance prototypes selected from $\mathbf{M}$ is determined by two parameters β and T. In our implementation, β is set to be 0.05, and T is the average of the maximal and minimal $\log(DD)$

values for all local maximizers found (line 8). These two parameters may need to be adjusted for other applications. However, empirical study shows that the performance of the classifier is not sensitive to β and T. Experimental analysis of the conditions under which the algorithm will find good instance prototypes is given in Section 4.5.

3.2.4 COMPUTING BAG FEATURES

Let $\{(\mathbf{x}_k^*, \mathbf{w}_k^*) : k = 1, \cdots, n\}$ be the collection of instance prototypes given by Algorithm 3.1. We define bag features, $\phi(\mathbf{B}_i)$, for a bag $\mathbf{B}_i = \{\mathbf{x}_{ij} : j = 1, \cdots, N_i\}$, as

$$\phi(\mathbf{B}_i) = \begin{bmatrix} \min_{j=1,\cdots,N_i} \|\mathbf{x}_{ij} - \mathbf{x}_1^*\|_{\mathbf{w}_1^*} \\ \min_{j=1,\cdots,N_i} \|\mathbf{x}_{ij} - \mathbf{x}_2^*\|_{\mathbf{w}_2^*} \\ \vdots \\ \min_{j=1,\cdots,N_i} \|\mathbf{x}_{ij} - \mathbf{x}_n^*\|_{\mathbf{w}_n^*} \end{bmatrix}. \tag{6}$$

In the definition (6), each bag feature is defined by one instance prototype and one instance from the bag, i.e., the instance that is "closest" to the instance prototype. A bag feature gives the smallest distance (or highest similarity score) between any instance in the bag and the corresponding instance prototype. Hence, it can also be viewed as a measure of the degree that an instance prototype shows up in the bag.

3.3 Comparing DD-SVM with MI-SVM

The following pseudo code summarizes the learning process of DD-SVM. The input is $\mathcal{D}$, a collection of bags with binary labels. The output is an SVM classifier defined by (2).

Algorithm 3.2 Learning DD-SVM

DD-SVM($\mathcal{D}$)

```
1 let S be the empty set
2 IP = MainLearnIPs(D)
3 FOR (every bag B in D)
4    define bag features φ(B) according to (6)
5    add (φ(B),y) to S where y is the label of B
6 END
7 train a standard SVM using S
8 OUTPUT (the SVM)
```

MI-SVM, proposed by Andrews et al. (2003), is also an SVM-based MIL method. In Section 4, we experimentally compare DD-SVM against MI-SVM. An algorithmic description of MI-SVM is given below. The input is a collection of labeled bags $\mathcal{D}$. The output is a classifier of the form

$$\text{label}(\mathbf{B}_i) = \text{sign}\left(max_{j=1,\cdots,N_i} f(\mathbf{x}_{ij})\right) \tag{7}$$

where $\mathbf{x}_{ij}$, $j = 1, \cdots, N_i$, are instances of $\mathbf{B}_i$, f is a function given by SVM learning.

Algorithm 3.3 Learning MI-SVM

MI-SVM($\mathcal{D}$)

```
 1 let P be the empty set
 2 FOR (every positive bag B in D)
 3    set x* be the average of instances in B
 4    add (x*, 1) to P
 5 END
 6 let N be the empty set
 7 FOR (every negative bag B in D)
 8    FOR (every instance x in B)
 9       add (x, −1) to N
10    END
11 END
12 REPEAT
13    set P' = P
14    set S = P' ∪ N
15    train a standard SVM, label(x) = sign(f(x)), using S
16    let P be the empty set
17    FOR (every positive bag B in D)
18       set x* = arg max_{x∈B} f(x)
19       add (x*, 1) to P
20    END
21 WHILE (P ≠ P')
22 OUTPUT (the classifier defined by (7))
```

In the above pseudo code for MI-SVM, the key steps are the loop given by lines 12–21. During each iteration, a standard SVM classifier, $\text{label}(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$, is trained in the instance space. The training set is the union of negative instances and positive instances. Negative instances are those from every negative bag. Each positive instance represents a positive bag. It is chosen to be the instance, in a positive bag, with the maximum f value from the previous iteration. In the first iteration, each positive instance is initialized to be the average of the feature vectors in the bag. The loop terminates if the set of positive instances selected for the next iteration is identical to that of the current iteration.

The crucial difference between DD-SVM and MI-SVM lies in the underlying assumption. MI-SVM method, as well as other standard MIL methods (such as the DD approach proposed by Maron and Lozano-Pérez, 1998), assumes that if a bag is labeled negative then all instances in that bag is negative, and if a bag is labeled positive, then as least one of the instances in that bag is a positive instance. In MI-SVM, one instance is selected to represent the whole positive bag. An SVM is trained in the instance feature space using all negative instances and the selected positive instances. Our DD-SVM method assumes that a positive bag must contain some number of instances satisfying various properties, which are captured by bag features. Each bag feature is defined by an instance in the bag and an instance prototype derived from the DD function. Hence, the bag features summarize the bag along several dimensions. An SVM is then trained in the bag feature space.

4. Experiments

We present systematic evaluations of DD-SVM based on a collection of images from the COREL and the MUSK data sets. The data sets and the source code of DD-SVM can be downloaded at `http://www.cs.uno.edu/~yixin/ddsvm.html`. Section 4.1 describes the experimental setup for image categorization, including the image data set, the implementation details, and the selection of parameters. Section 4.2 compares DD-SVM with MI-SVM and color histogram-based SVM using COREL data. The effect of inaccurate image segmentation on classification accuracies is demonstrated in Section 4.3. Section 4.4 illustrates the performance variations when the number of image categories increases. Analysis of the effects of training sample size and diversity of images is given in Section 4.5. Results on the MUSK data sets are presented in Section 4.6. Computational issues are discussed in Section 4.7.

4.1 Experimental Setup for Image Categorization

The image data set employed in our empirical study consists of $2,000$ images taken from 20 CD-ROMs published by COREL Corporation. Each COREL CD-ROM of 100 images represents one distinct topic of interest. Therefore, the data set has 20 thematically diverse image categories, each containing 100 images. All the images are in JPEG format with size 384×256 or 256×384. We assigned a keyword (or keywords) to describe each image category. The category names and some randomly selected sample images from each category are shown in Figure 3.

Images within each category are randomly divided into a training set and a test set each with 50 images. We repeat each experiment for 5 random splits, and report the average of the results obtained over 5 different test sets together with the 95% confidence interval. The SVM^{Light} (Joachims, 1999) software is used to train the SVMs. The classification problem here is clearly a multi-class problem. We use the one-against-the-rest approach: (a) for each category, an SVM is trained to separate that category from all the other categories; (b) the final predicted class label is decided by the winner of all SVMs, i.e., one with the maximum value inside the $\text{sign}(\cdot)$ function in (2).

Two other image classification methods are implemented for comparison. One is a histogram-based SVM classification approach proposed by Chapelle et al. (1999). We denote it by Hist-SVM. Each image is represented by a color histogram in the LUV color space. The dimension of each histogram is 125. The other is MI-SVM (Andrews et al., 2003). MI-SVM uses the same set of region features as our approach (it is implemented according to the pseudo code in Algorithm 3.3). The learning problems in Hist-SVM and MI-SVM are solved by SVM^{Light}. The Gaussian kernel, $K(\mathbf{x},\mathbf{z}) = e^{-s\|\mathbf{x}-\mathbf{z}\|^2}$, is used in all three methods.

Several parameters need to be specified for SVM^{Light}.[3] The most significant ones are s and C (the constant in (1) controlling the trade-off between training error and regularization). We apply the following strategy to select these two parameters: We allow each one of the two parameters be respectively chosen from two sets each containing 10 predetermined numbers. For every pair of values of the two parameters (there are 100 pairs in total), a twofold cross-validation error on the training set is recorded. The pair that gives the minimum twofold cross-validation error is selected to be the "optimal" parameters. Note that the above procedure is applied only once for each method. Once the parameters are determined, they are used in all subsequent image categorization experiments.

3. SVM^{Light} software and detailed descriptions of all its parameters are available at `http://svmlight.joachims.org`.

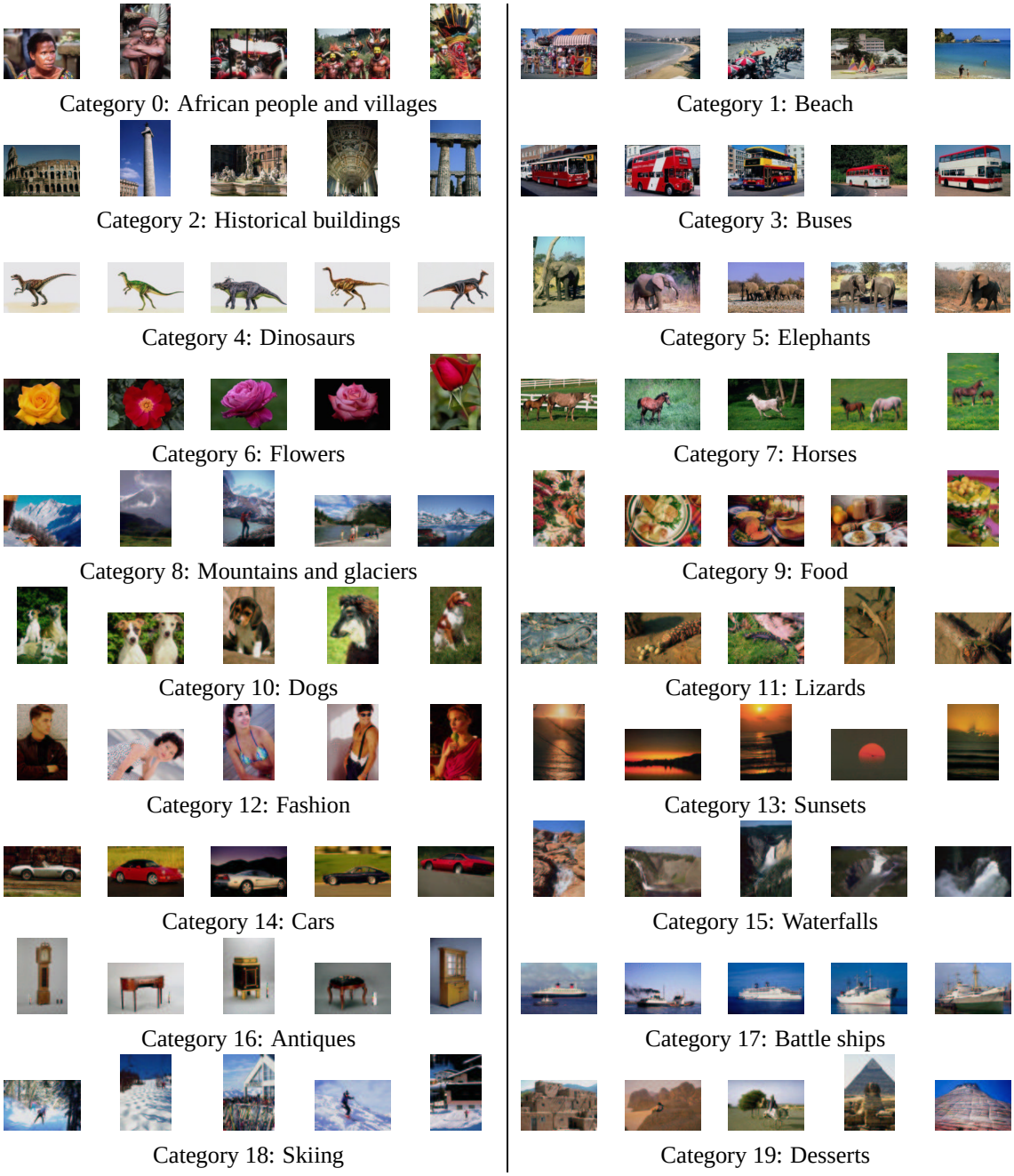


Figure 3: Sample images taken from 20 image categories.

4.2 Categorization Results

The classification results provided in Table 1 are based on images in Category 0 to Category 9, i.e., 1,000 images. Results for the whole data set will be given in Section 4.4. DD-SVM performs much better than Hist-SVM with a 14.8% difference in average classification accuracy. Compared with MI-SVM, the average accuracy of DD-SVM is 6.8% higher. As we will see in Section 4.4,

	Average Accuracy : [95% confidence interval]
DD-SVM	81.5% : [78.5%, 84.5%]
Hist-SVM	66.7% : [64.5%, 68.9%]
MI-SVM	74.7% : [74.1%, 75.3%]

Table 1: Image categorization performance of DD-SVM, Hist-SVM, and MI-SVM. The numbers listed are the average classification accuracies over 5 random test sets and the corresponding 95% confidence intervals. The images belong to Category 0 to Category 9. Training and test sets are of equal size.

	Cat. 0	Cat. 1	Cat. 2	Cat. 3	Cat. 4	Cat. 5	Cat. 6	Cat. 7	Cat. 8	Cat. 9
Cat. 0	**67.7**%	3.7%	5.7%	0.0%	0.3%	8.7%	5.0%	1.3%	0.3%	7.3%
Cat. 1	1.0%	**68.4**%	4.3%	4.3%	0.0%	3.0%	1.3%	1.0%	15.0%	1.7%
Cat. 2	5.7%	5.0%	**74.3**%	2.0%	0.0%	3.3%	0.7%	0.0%	6.7%	2.3%
Cat. 3	0.3%	3.7%	1.7%	**90.3**%	0.0%	0.0%	0.0%	0.0%	1.3%	2.7%
Cat. 4	0.0%	0.0%	0.0%	0.0%	**99.7**%	0.0%	0.0%	0.0%	0.0%	0.3%
Cat. 5	5.7%	3.3%	6.3%	0.3%	0.0%	**76.0**%	0.7%	4.7%	2.3%	0.7%
Cat. 6	3.3%	0.0%	0.0%	0.0%	0.0%	1.7%	**88.3**%	2.3%	0.7%	3.7%
Cat. 7	2.3%	0.3%	0.0%	0.0%	0.0%	2.0%	1.0%	**93.4**%	0.7%	0.3%
Cat. 8	0.3%	15.7%	5.0%	1.0%	0.0%	4.3%	1.0%	0.7%	**70.3**%	1.7%
Cat. 9	3.3%	1.0%	0.0%	3.0%	0.7%	1.3%	1.0%	2.7%	0.0%	**87.0**%

Table 2: The confusion matrix of image categorization experiments (over 5 randomly generated test sets). Each row lists the average percentage of images (test images) in one category classified to each of the 10 categories by DD-SVM. Numbers on the diagonal show the classification accuracy for each category.

the difference becomes even greater as the number of categories increases. This suggests that the proposed method is more effective than MI-SVM in learning concepts of image categories under the same image representation. The MIL formulation of our method may be better suited for region-based image classification than that of MI-SVM.

Next, we make a closer analysis of the performance by looking at classification results on every category in terms of the confusion matrix. The results are listed in Table 2. Each row lists the average percentage of images in one category classified to each of the 10 categories by DD-SVM. The numbers on the diagonal show the classification accuracy for each category, and off-diagonal entries indicate classification errors. Ideally, one would expect the diagonal terms be all 1's, and the off-diagonal terms be all 0's. A detailed examination of the confusion matrix shows that two of the largest errors (the underlined numbers in Table 2) are errors between Category 1 (Beach) and Category 8 (Mountains and glaciers): 15.0% of "Beach" images are misclassified as "Mountains and glaciers;" 15.7% of "Mountains and glaciers" images are misclassified as "Beach." Figure 4 presents 12 misclassified images (in at least one experiment) from both categories. All "Beach" images in Figure 4 contain mountains or mountain-like regions, while all the "Mountains and glaciers" images

Figure 4: Some sample images taken from two categories: "Beach" and "Mountains and glaciers." All the listed "Beach" images are misclassified as "Mountains and glaciers," while the listed "Mountains and glaciers" images are misclassified as "Beach."

have regions corresponding to river, lake, or even ocean. In other words, although these two image categories do not share annotation words, they are semantically related and visually similar. This may be the reason for the classification errors.

4.3 Sensitivity to Image Segmentation

Because image segmentation cannot be perfect, being robust to segmentation-related uncertainties becomes a critical performance index for a region-based image classification method. Figure 5 shows two images, "African people" and "Horses," and the segmentation results with different numbers of regions (the results are obtained by varying the stopping criteria of the k-mean segmentation algorithm presented in Section 2). Regions are shown in their representative colors. We can see from Figure 5 that, under some stopping criteria, objects totally different in semantics may be clustered into the same region (under-segmented). While under some other stopping criteria, one object may be divided into several regions (over-segmented).

In this section, we compare the performance of DD-SVM with MI-SVM when the coarseness of image segmentation varies. To give a fair comparison, we control the coarseness of image segmentation by adjusting the stopping criteria of the k-means segmentation algorithm. We pick 5 different stopping criteria. The corresponding average numbers of regions per image (computed over $1,000$ images from Category 0 to Category 9) are 4.31, 6.32, 8.64, 11.62, and 12.25. The average classification accuracies (over 5 randomly generated test sets) under each coarseness level and the corresponding 95% confidence intervals are presented in Figure 6.

The results in Figure 6 indicate that DD-SVM outperforms MI-SVM on all 5 coarseness levels. In addition, for DD-SVM, there are no significant changes in the average classification accuracy for different coarseness levels. While the performance of MI-SVM degrades as the average number of regions per image increases. The difference in average classification accuracies between the two methods are 6.8%, 9.5%, 11.7%, 13.8%, and 27.4% as the average number of regions per image increases. This appears to support the claim that DD-SVM has low sensitivity to image segmentation.

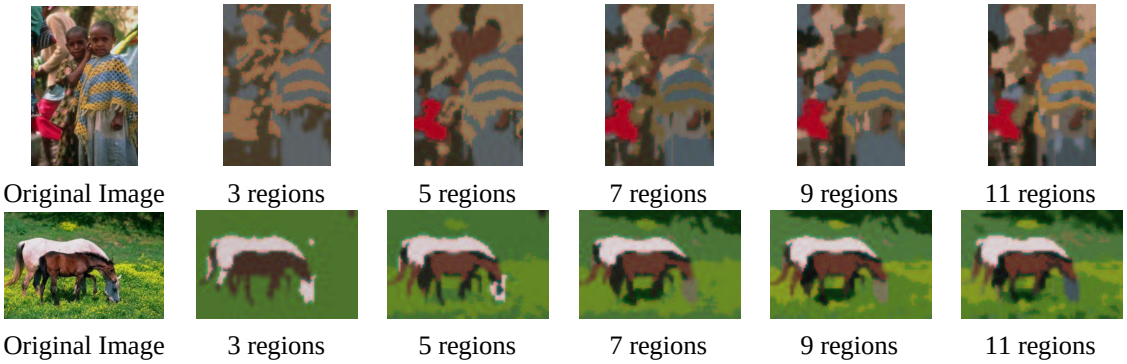


Figure 5: Segmentation results given by the *k*-means clustering algorithm with 5 different stopping criteria. Original images, which are taken from "African people" and "Horses" categories, are in the first column. Segmented regions are shown in their representative colors.

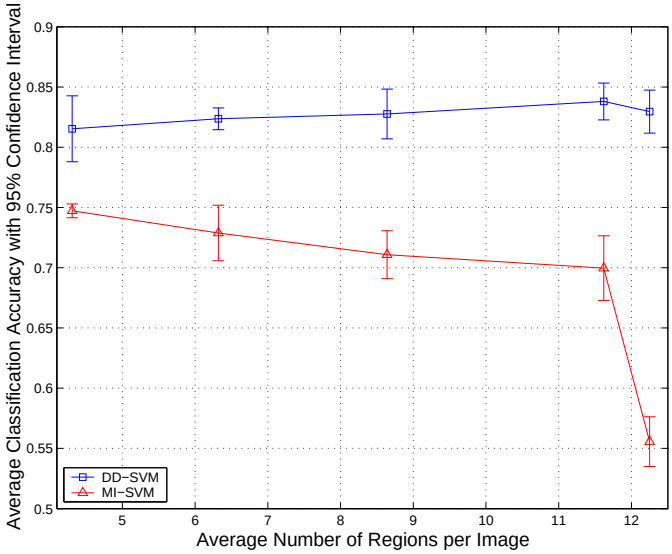


Figure 6: Comparing DD-SVM with MI-SVM on the robustness to image segmentation. The experiment is performed on 1,000 images in Category 0 to Category 9 (training and test sets are of equal size). The average classification accuracies and the corresponding 95% confidence intervals are computed over 5 randomly generated test sets. The average numbers of regions per image are 4.31, 6.32, 8.64, 11.62, and 12.25.

4.4 Sensitivity to the Number of Categories in a Data Set

Although the experimental results in Section 4.2 and 4.3 demonstrate the good performance of DD-SVM using 1,000 images in Category 0 to Category 9, the scalability of the method remains a question: how does the performance scale as the number of categories in a data set increases? We attempt to empirically answer this question by performing image categorization experiments over

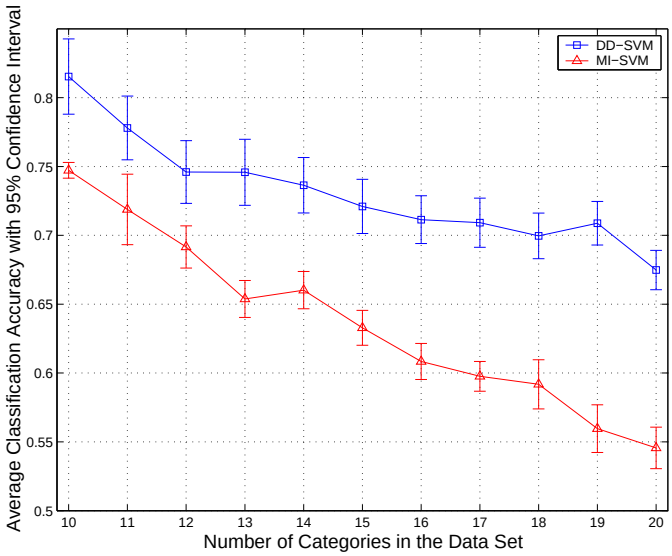


Figure 7: Comparing DD-SVM with MI-SVM on the robustness to the number of categories in a data set. The experiment is performed on 11 different data sets. The number of categories in a data set varies from 10 to 20. A data set with i categories contains $100 \times i$ images from Category 0 to Category $i-1$ (training and test sets are of equal size). The average classification accuracies and the corresponding 95% confidence intervals are computed over 5 randomly generated test sets.

data sets with different numbers of categories. A total of 11 data sets are used in the experiments. The number of categories in a data set varies from 10 to 20. A data set with i categories contains $100 \times i$ images from Category 0 to Category $i-1$. The average classification accuracies over 5 randomly generated test sets and the corresponding 95% confidence intervals are presented in Figure 7. We also include the results of MI-SVM for comparison.

We observe a decrease in average classification accuracy as the number of categories increases. When the number of categories becomes doubled (increasing from 10 to 20 categories), the average classification accuracy of DD-SVM drops from 81.5% to 67.5%. However, DD-SVM seems to be less sensitive to the number of categories in a data set than MI-SVM. This is indicated, in Figure 8, by the difference in average classification accuracies between the two methods as the number of categories in a data set increases. It should be clear that our method outperforms MI-SVM consistently. And the performance discrepancy increases with the number of categories. For the 1000-image data set with 10 categories, the difference is 6.8%. This number is nearly doubled (12.9%) when the number of categories becomes 20. In other words, the performance degradation of DD-SVM is slower than that of MI-SVM as the number of categories increases.

4.5 Sensitivity to the Size and Diversity of Training Images

We test the sensitivity of DD-SVM to the size of training set using $1,000$ images from Category 0 to Category 9 with the size of the training sets being 100, 200, 300, 400, and 500 (the number of images from each category is $\frac{size\ of\ the\ training\ set}{10}$). The corresponding numbers of test images are 900,

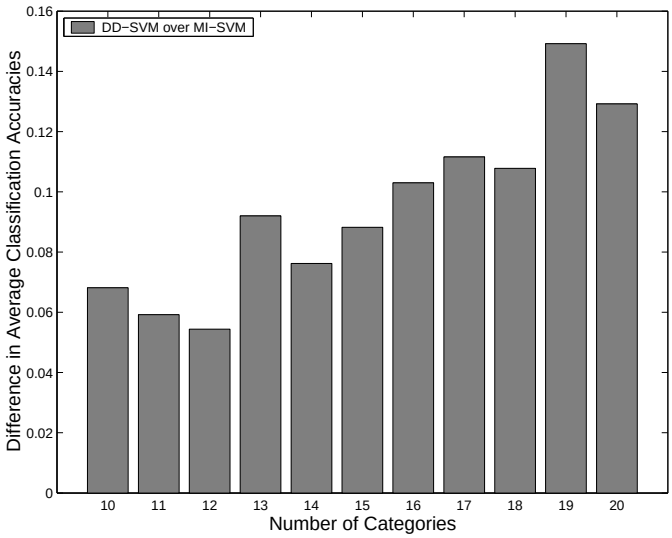


Figure 8: Difference in average classification accuracies between DD-SVM and MI-SVM as the number of categories varies. A positive number indicates that DD-SVM has higher average classification accuracy.

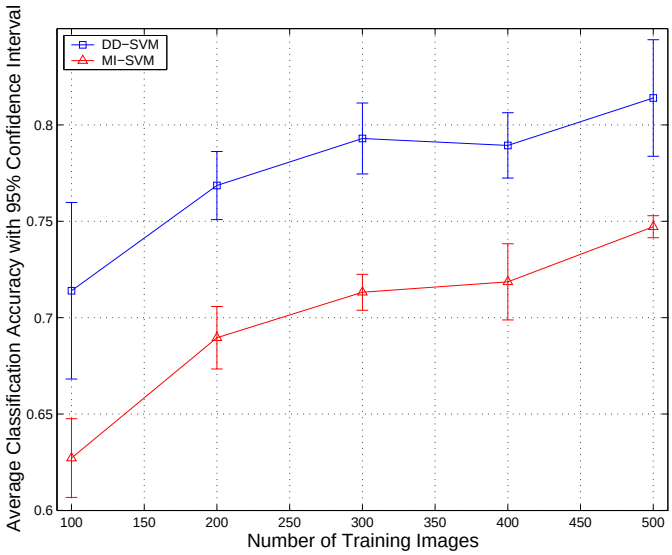


Figure 9: Comparing DD-SVM with MI-SVM as the number of training images varies from 100 to 500. The experiment is performed on $1,000$ images in Category 0 to Category 9. The average classification accuracies and the corresponding 95% confidence intervals are computed over 5 randomly generated test sets.

800, 700, 600, and 500. As indicated in Figure 9, when the number of training images decreases, the average classification accuracy of DD-SVM degrades as expected. Figure 9 also shows that the

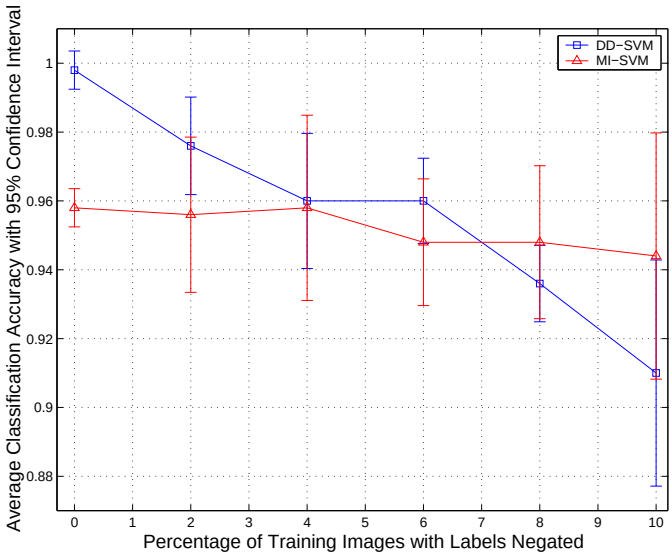


Figure 10: Comparing DD-SVM with MI-SVM as the diversity of training images varies. The experiment is performed on 200 images in Category 2 (Historical buildings) and Category 7 (Horses). The average classification accuracies and the corresponding 95% confidence intervals are computed over 5 randomly generated test sets. Training and test sets have equal size.

performance of DD-SVM degrades in roughly the same speed as that of MI-SVM: the differences in average classification accuracies between DD-SVM and MI-SVM are 8.7%, 7.9%, 8.0%, 7.1%, and 6.8% when the training sample size varies from 100 to 500.

To test the performance of DD-SVM as the diversity of training images varies, we need to define a measure of the diversity. In terms of binary classification, we define the diversity of images as a measure of the number of positive images that are "similar" to negative images and the number of negative images that are "similar" to positive images. In this experiment, training sets with different diversities are generated as follows. We first randomly pick d% of positive images and d% of negative images from a training set. Then, we modify the labels of the selected images by negating their labels, i.e., positive (negative) images become negative (positive) images. Finally, we put these images with new labels back to the training set. The new training set has d% of images with negated labels. It should be clear that $d = 0$ and $d = 50$ correspond to the lowest and highest diversities, respectively.

We compare DD-SVM with MI-SVM for $d = 0$, 2, 4, 6, 8, and 10 based on 200 images from Category 2 (Historical buildings) and Category 7 (Horses). The training and test sets have equal size. The average classification accuracies (over 5 randomly generated test sets) and the corresponding 95% confidence intervals are presented in Figure 10. We observe that the average classification accuracy of DD-SVM is about 4% higher than that of MI-SVM when $d = 0$. And this difference is statistically significant. However, if we randomly negate the labels of one positive image and one negative image in the training set (i.e., $d = 2$ in this experimental setup), the performance of DD-SVM is roughly the same as that of MI-SVM: although DD-SVM still leads MI-SVM by 2% of

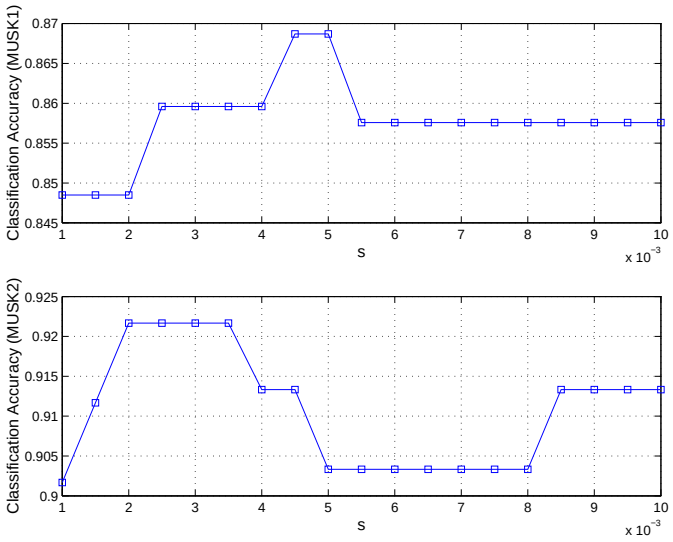


Figure 11: Average accuracy of 10-fold cross-validation on MUSK data sets using DD-SVM. The parameters are $C = 1,000$ and s taking 19 values evenly distributed in $[0.001, 0.01]$.

average classification accuracy, the difference is statistically-indistinguishable. As d increases, DD-SVM and MI-SVM generate roughly the same performance. This suggests that DD-SVM is more sensitive to the diversity of training images than MI-SVM. We attempt to explain this observation as follows. The DD function (3) used in Algorithm 3.1 is very sensitive to instances in negative bags. It is not difficult to derive from (3) that the DD value at a point is substantially reduced is there is a single instance from negative bags close to the point. Therefore, negating labels of one positive and one negative image could significantly modify the DD function, and consequently, the instance prototypes learned by Algorithm 3.1.

4.6 MUSK Data Sets

The MUSK data sets, MUSK1 and MUSK2 (Blake and Merz, 1998), are benchmark data sets for MIL. Both data sets consist of descriptions of molecules. Specifically, a bag represents a molecule. Instances in a bag represent low-energy conformations of the molecule. Each instance (or conformation) is defined by a 166-dimensional feature vector describing the surface of a low-energy conformation. The data were preprocessed by dividing each feature value by 100. This was done so that learning of instance prototypes would not begin at a flat area of the instance space. MUSK1 has 92 molecules (bags), of which 47 are labeled positive, with an average of 5.17 conformations (instances) per molecule. MUSK2 has 102 molecules, of which 39 are positive, with an average of 64.69 conformations per molecule.

Figure 11 shows the average accuracy of 10-fold cross-validation using DD-SVM with $C = 1,000$ and the s parameter of the Gaussian kernel taking the following values: 0.001, 0.0015, 0.002, 0.0025, 0.003, 0.0035, 0.004, 0.0045, 0.005 0.0055, 0.006, 0.0065, 0.007, 0.0075, 0.008, 0.0085, 0.009, 0.0095, 0.01. As a function of s, the average 10-fold cross-validation accuracy of DD-SVM varies within $[84.9\%, 86.9\%]$ (MUSK1) or $[90.2\%, 92.2\%]$ (MUSK2). For both data sets, the me-

	Average Accuracy	
	MUSK1	MUSK2
DD-SVM	85.8%	**91.3**%
IAPR	**92.4**%	89.2%
DD	88.9%	82.5%
EM-DD	84.8%	84.9%
MI-SVM	77.9%	84.3%
mi-SVM	87.4%	83.6%
MI-NN	88.0%	82.0%
Multinst	76.7%	84.0%

Table 3: Comparison of averaged 10-fold cross-validation accuracies on MUSK data sets.

dian of the average accuracy, which is robust over a range of parameter values, is reported in Table 3. Table 3 also summarizes the performance of seven MIL algorithms in the literature: IAPR (Dietterich et al., 1997), DD (Maron and Lozano-Pérez, 1998), EM-DD (Zhang and Goldman, 2002),[4] MI-SVM and mi-SVM (Andrews et al., 2003), MI-NN (Ramon and De Raedt, 2000), and Multinst (Auer, 1997). Although DD-SVM is outperformed by IAPR, DD, MI-NN, and mi-SVM on MUSK1, it generates the best performance on MUSK2. Overall, DD-SVM achieves very competitive accuracy values.

4.7 Speed

On average, the learning of each binary classifier using a training set of 500 images (4.31 regions per image) takes around 40 minutes of CPU time on a Pentium III 700MHz PC running the Linux operating system. Algorithm 3.1 is implemented in Matlab with the quasi-Newton search procedure written in the C programming language. Among this amount of time, the majority is spent on learning instance prototypes, in particular, the `FOR` loop of **LearnIPs($\mathcal{D}$)** in Algorithm 3.1. This is because the quasi-Newton search needs to be applied with every instance in every positive bag as starting points (each optimization only takes a few seconds). However, since these optimizations are independent of each other, they can be fully parallelized. Thus the training time may be reduced significantly.

5. Conclusions and Future Work

In this paper, we proposed a region-based image categorization method using an extension of Multiple-Instance Learning, DD-SVM. Each image is represented as a collection of regions obtained from image segmentation using the *k*-means algorithm. In DD-SVM, each image is mapped to a point in a bag feature space, which is defined by a set of instance prototypes learned with the Diverse Density function. SVM-based image classifiers are then trained in the bag feature space. We demonstrate that DD-SVM outperforms two other methods in classifying images from 20 distinct semantic classes. In addition, DD-SVM generates highly competitive results on the MUSK data sets, which are benchmark data sets for MIL.

4. The EM-DD results reported in Zhang and Goldman (2002) were obtained by selecting the optimal solution using the test data. The EM-DD result cited in this paper is provide by Andrews et al. (2003) using the correct algorithm.

The proposed image categorization method has several limitations:

- The semantic meaning of an instance prototype is usually unknown because the learning algorithm in Section 3 does not associate a linguistic label with each instance prototype. As a result, "region naming" (Barnard et al., 2003) is not supported by DD-SVM.

- It may not be possible to learn certain concepts through the method. For example, texture images can be designed using a simple object (or region), such as a T-shaped object. By varying orientation, frequency of appearance, and alignment of the object, one can get texture images that are visually different. In other words, the concept of texture depends on not only the individual object but also the spatial relationship of objects (or instances). But this spatial information is not exploited by the current work. As pointed out by one reviewer of the initial draft, a possible way to tackle this problem is to use Markov random field type of models (Modestino and Zhang, 1992).

The performance of image categorization may be improved in the following ways:

- The image segmentation algorithm may be improved. The current k-means algorithm is relatively simple and efficient. But over-segmentation and under-segmentation may happen frequently for a fixed stopping criterion. Although the empirical results in Section 4.3 show that the proposed method has low sensitivity to image segmentation, a semantically more accurate segmentation algorithm may improve the overall classification accuracy.

- The definition of the DD function may be improved. The current DD function, which is a multiplicative model, is very sensitive to instances in negative bags. It can be easily observed from (3) that the DD value at a point is significantly reduced if there is a single instance from a negative bag close to the point. This property may be desirable for some applications, such as drug discovery (Maron and Lozano-Pérez, 1998), where the goal is to learn a single point in the instance feature space with the maximum DD value from an almost "noise free" data set. But this is not a typical problem setting for region-based image categorization where data usually contain noise. Thus a more robust definition of DD, such as an additive model, may enhance the performance.

As pointed out by a reviewer of the initial draft, scene category can be a vector. For example, a scene can be $\{$*mountain*, *beach*$\}$ in one dimension, but also $\{$*winter*, *summer*$\}$ in the other dimension. Under this scenario, our current work can be applied in two ways: (a) design a multi-class classifier for each dimension, i.e., *mountain*/*beach* classifier for one dimension and *winter*/*summer* classifier for the other dimension; or (b) design one multi-class classifier taking all scene categories into consideration, i.e., *mountain-winter*, *mountain-summer*, *beach-winter*, and *beach-summer* categories.

In our experimental evaluations, image semantic categories are assumed to be well-defined. As pointed out by one of the reviewers, image semantics is inherently linguistic, therefore, can only be defined loosely. Thus a methodologically well-defined evaluation technique should take into account scenarios with differing amounts of knowledge about the image semantics. Unless this issue can be fully investigated, our image categorization results should be interpreted cautiously.

As continuations of this work, several directions may be pursued. The proposed method can potentially be applied to automatically index images using linguistic descriptions. It can also be

integrated to content-based image retrieval systems to group images into semantically meaningful categories so that semantically-adaptive searching methods applicable to each category can be applied. The current instance prototype learning scheme may be improved by boosting techniques. Art and biomedical images would be interesting applications.

Acknowledgments

The material is based upon work supported by the National Science Foundation under Grant No. IIS-0219272 and CNS-0202007, The Pennsylvania State University, University of New Orleans, The Research Institute for Children, the PNC Foundation, SUN Microsystems under Grant EDUD-7824-010456-US, and NASA/EPSCoR DART Grant NCC5-573. The authors would like to thank Jia Li for making many suggestions on the initial manuscript. We thank the reviewers for valuable suggestions. We would also like to thank Jinbo Bi, Seth Pincus, Andrés Castaño, C. Lee Giles, Donald Richards, and John Yen for helpful discussions.

References

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, pages 561–568. Cambridge, MA:MIT Press, 2003.

P. Auer. On learning from mult-instance examples: empirical evaluation of a theoretical approach. In *Proc. 14th Int'l Conf. on Machine Learning*, pages 21–29, 1997.

K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

K. Barnard and D. Forsyth. Learning the semantics of words and pictures. In *Proc. 8th Int'l Conf. on Computer Vision*, pages II:408–415, 2001.

C. L. Blake and C. J. Merz. UCI Repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

A. Blum and A. Kalai. A note on learning from multiple-instance examples. *Machine Learning*, 30 (1):23–29, 1998.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

Y. Chen and J. Z. Wang. A region-based fuzzy feature matching approach to content-based image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1252–1267, 2002.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

I. Daubechies. *Ten Lectures on Wavelets*. Capital City Press, 1992.

T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, 1999.

A. Gersho. Asymptotically optimum block quantization. *IEEE Transactions on Information Theory*, 25(4):373–380, 1979.

M. M. Gorkani and R. W. Picard. Texture orientation for sorting photos 'at a glance'. In *Proc. 12th Int'l Conf. on Pattern Recognition*, pages I:459–464, 1994.

J. A. Hartigan and M. A. Wong. Algorithm AS136: A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

J. Huang, S. R. Kumar, and R. Zabih. An automatic hierarchical image classification scheme. In *Proc. 6th ACM Int'l Conf. on Multimedia*, pages 219–228, 1998.

T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. Edited by B. Schölkopf, C. J.C. Burges, and A.J. Smola, Cambridge, MA: MIT Press, 1999.

J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1075–1088, 2003.

W. Y. Ma and B. Manjunath. NeTra: A toolbox for navigating large image databases. In *Proc. IEEE Int'l Conf. on Image Processing*, pages 568–571, 1997.

O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 10*, pages 570–576. Cambridge, MA: MIT Press, 1998.

O. Maron and A. L. Ratan. Multiple-instance learning for natural scene classification. In *Proc. 15th Int'l Conf. on Machine Learning*, pages 341–349, 1998.

D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W H Freeman & Co., 1983.

J. W. Modestino and J. Zhang. A Markov random field model-based approach to image interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):606–615, 1992.

K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the trees: a graphical model relating features, objects, and scenes. In *Advances in Neural Information Processing Systems 16*. Cambridge, MA:MIT Press, 2004.

S. A. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The art of scientific computing*. second edition, Cambridge University Press, New York, 1992.

J. Ramon and L. De Raedt. Multi instance neural networks. In *Proc. ICML-2000 Workshop on Attribute-Value and Relational Learning*, 2000.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

J. R. Smith and C.-S. Li. Image classification and querying using composite region templates. *Int'l J. Computer Vision and Image Understanding*, 75(1/2):165–174, 1999.

T. M. Strat. *Natural Object Recognition*. Berlin: Springer-Verlag, 1992.

M. Szummer and R. W. Picard. Indoor-outdoor image classification. In *Proc. IEEE Int'l Workshop on Content-Based Access of Image and Video Databases*, pages 42–51, 1998.

M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, 1995.

A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H.-J. Zhang. Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10(1):117–130, 2001.

N. Vasconcelos and A. Lippman. A Bayesian framework for semantic content characterization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 566–571, 1998.

J. Z. Wang, J. Li, R. M. Gray, and G. Wiederhold. Unsupervised multiresolution segmentation for images with low depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):85–91, 2001a.

J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001b.

C. Yang and T. Lozano-Pérez. Image database retrieval with multiple-instance learning techniques. In *Proc. IEEE Int'l Conf. on Data Engineering*, pages 233–243, 2000.

H. Yu and W. Wolf. Scenic classification methods for image and video databases. In *Proc. SPIE Int'l Conf. on Digital Image Storage and archiving systems*, pages 2606:363–371, 1995.

Q. Zhang and S. A. Goldman. EM-DD: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems 14*, pages 1073–1080. Cambridge, MA: MIT Press, 2002.

Q. Zhang, S. A. Goldman, W. Yu, and J. Fritts. Content-based image retrieval using multiple-instance learning. In *Proc. 19th Int'l Conf. on Machine Learning*, pages 682–689, 2002.

S. C. Zhu and A. Yuille. Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900, 1996.

Journal of Machine Learning Research 5 (2004) 941–973 Submitted 5/03; Revised 10/03; Published 8/04

Boosting as a Regularized Path to a Maximum Margin Classifier

Saharon Rosset SROSSET@US.IBM.COM
Data Analytics Research Group
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA

Ji Zhu JIZHU@UMICH.EDU
Department of Statistics
University of Michigan
Ann Arbor, MI 48109, USA

Trevor Hastie HASTIE@STAT.STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305,USA

Editor: Robert Schapire

Abstract

In this paper we study boosting methods from a new perspective. We build on recent work by Efron et al. to show that boosting approximately (and in some cases exactly) minimizes its loss criterion with an l_1 constraint on the coefficient vector. This helps understand the success of boosting with early stopping as regularized fitting of the loss criterion. For the two most commonly used criteria (exponential and binomial log-likelihood), we further show that as the constraint is relaxed—or equivalently as the boosting iterations proceed—the solution converges (in the separable case) to an "l_1-optimal" separating hyper-plane. We prove that this l_1-optimal separating hyper-plane has the property of maximizing the minimal l_1-margin of the training data, as defined in the boosting literature. An interesting fundamental similarity between boosting and kernel support vector machines emerges, as both can be described as methods for regularized optimization in high-dimensional predictor space, using a computational trick to make the calculation practical, and converging to margin-maximizing solutions. While this statement describes SVMs exactly, it applies to boosting only approximately.

Keywords: boosting, regularized optimization, support vector machines, margin maximization

1. Introduction and Outline

Boosting is a method for iteratively building an additive model

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_{j_t}(\mathbf{x}), \tag{1}$$

where $h_{j_t} \in \mathcal{H}$—a large (but we will assume finite) dictionary of candidate predictors or "weak learners"; and h_{j_t} is the basis function selected as the "best candidate" to modify the function at stage t. The model F_T can equivalently be represented by assigning a coefficient to each dictionary

function $h \in \mathcal{H}$ rather than to the selected h_{j_t}'s only:

$$F_T(\mathbf{x}) = \sum_{j=1}^{J} h_j(\mathbf{x}) \cdot \beta_j^{(T)}, \tag{2}$$

where $J = |\mathcal{H}|$ and $\beta_j^{(T)} = \sum_{j_t=j} \alpha_t$. The "$\beta$" representation allows us to interpret the coefficient vector $\beta^{(T)}$ as a vector in $\mathcal{R}^J$ or, equivalently, as the hyper-plane which has $\beta^{(T)}$ as its normal. This interpretation will play a key role in our exposition.

Some examples of common dictionaries are:

- The training variables themselves, in which case $h_j(\mathbf{x}) = x_j$. This leads to our "additive" model F_T being just a linear model in the original data. The number of dictionary functions will be $J = d$, the dimension of $\mathbf{x}$.
- Polynomial dictionary of degree p, in which case the number of dictionary functions will be $J = \begin{pmatrix} p+d \\ d \end{pmatrix}$.
- Decision trees with up to k terminal nodes, if we limit the split points to data points (or midway between data points as CART does). The number of possible trees is bounded from above (trivially) by $J \leq (np)^k \cdot 2^{k^2}$. Note that regression trees do not fit into our framework, since they will give $J = \infty$.

The boosting idea was first introduced by Freund and Schapire (1995), with their AdaBoost algorithm. AdaBoost and other boosting algorithms have attracted a lot of attention due to their great success in data modeling tasks, and the "mechanism" which makes them work has been presented and analyzed from several perspectives. Friedman et al. (2000) develop a statistical perspective, which ultimately leads to viewing AdaBoost as a gradient-based incremental search for a good additive model (more specifically, it is a "coordinate descent" algorithm), using the exponential loss function $C(y,F) = \exp(-yF)$, where $y \in \{-1,1\}$. The gradient boosting (Friedman, 2001) and anyboost (Mason et al., 1999) generic algorithms have used this approach to generalize the boosting idea to wider families of problems and loss functions. In particular, Friedman et al. (2000) have pointed out that the binomial log-likelihood loss $C(y,F) = \log(1+\exp(-yF))$ is a more natural loss for classification, and is more robust to outliers and misspecified data.

A different analysis of boosting, originating in the machine learning community, concentrates on the effect of boosting on the margins $y_i F(\mathbf{x}_i)$. For example, Schapire et al. (1998) use margin-based arguments to prove convergence of boosting to perfect classification performance on the training data under general conditions, and to derive bounds on the generalization error (on future, unseen data).

In this paper we combine the two approaches, to conclude that gradient-based boosting can be described, in the separable case, as an approximate margin maximizing process. The view we develop of boosting as an approximate path of optimal solutions to regularized problems also justifies early stopping in boosting as specifying a value for "regularization parameter".

We consider the problem of minimizing non-negative convex loss functions (in particular the exponential and binomial log-likelihood loss functions) over the training data, with an l_1 bound on the model coefficients:

$$\hat{\beta}(c) = \arg\min_{\|\beta\|_1 \leq c} \sum_i C(y_i, h(\mathbf{x}_i)'\beta). \tag{3}$$

Where $h(\mathbf{x}_i) = [h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_J(\mathbf{x}_i)]'$ and $J = |\mathcal{H}|$.[1]

Hastie et al. (2001, Chapter 10) have observed that "slow" gradient-based boosting (i.e., we set $\alpha_t = \varepsilon \ , \forall t$ in (1), with ε small) tends to follow the penalized path $\hat{\beta}(c)$ as a function of c, under some mild conditions on this path. In other words, using the notation of (2), (3), this implies that $\|\beta^{(c/\varepsilon)} - \hat{\beta}(c)\|$ vanishes with ε, for all (or a wide range of) values of c. Figure 1 illustrates this equivalence between ε-boosting and the optimal solution of (3) on a real-life data set, using squared error loss as the loss function. In this paper we demonstrate this equivalence further and formally

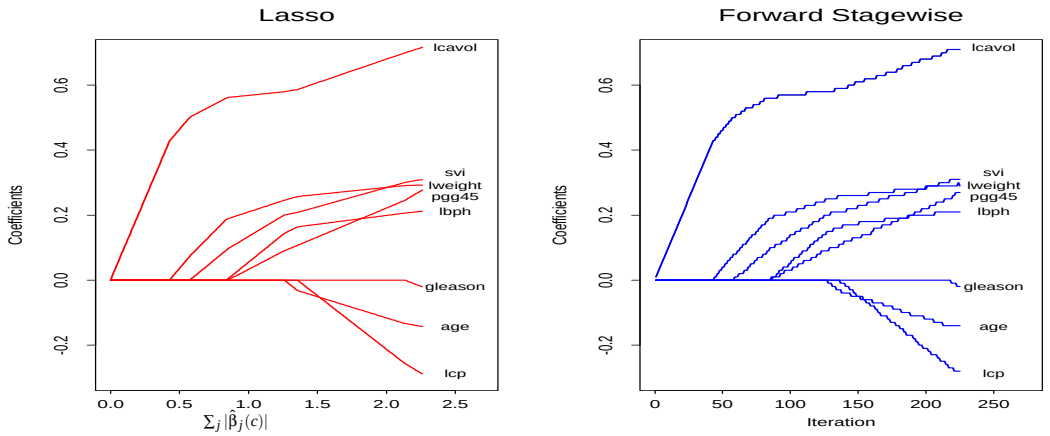


Figure 1: Exact coefficient paths(left) for l_1-constrained squared error regression and "boosting" coefficient paths (right) on the data from a prostate cancer study

state it as a conjecture. Some progress towards proving this conjecture has been made by Efron et al. (2004), who prove a weaker "local" result for the case where C is squared error loss, under some mild conditions on the optimal path. We generalize their result to general convex loss functions.

Combining the empirical and theoretical evidence, we conclude that boosting can be viewed as an approximate incremental method for following the l_1-regularized path.

We then prove that in the separable case, for both the exponential and logistic log-likelihood loss functions, $\hat{\beta}(c)/c$ converges as $c \to \infty$ to an "optimal" separating hyper-plane $\hat{\beta}$ described by

$$\hat{\beta} = \arg\max_{\|\beta\|_1=1} \min_i y_i \beta' h(\mathbf{x}_i). \tag{4}$$

In other words, $\hat{\beta}$ *maximizes the minimal margin* among all vectors with l_1-norm equal to 1.[2] This result generalizes easily to other l_p-norm constraints. For example, if $p = 2$, then $\hat{\beta}$ describes the optimal separating hyper-plane in the Euclidean sense, i.e., the same one that a non-regularized support vector machine would find.

Combining our two main results, we get the following characterization of boosting:

1. Our notation assumes that the minimum in (3) is unique, which requires some mild assumptions. To avoid notational complications we use this slightly abusive notation throughout this paper. In Appendix B we give explicit conditions for uniqueness of this minimum.
2. The margin maximizing hyper-plane in (4) may not be unique, and we show that in that case the limit $\hat{\beta}$ is still defined and it also maximizes the second minimal margin. See Appendix B.2 for details.

> ε-Boosting can be described as a gradient-descent search, approximately following the path of l_1-constrained optimal solutions to its loss criterion, and converging, in the separable case, to a "margin maximizer" in the l_1 sense.

Note that boosting with a large dictionary $\mathcal{H}$ (in particular if $n < J = |\mathcal{H}|$) guarantees that the data will be separable (except for pathologies), hence separability is a very mild assumption here.

As in the case of support vector machines in high dimensional feature spaces, the non-regularized "optimal" separating hyper-plane is usually of theoretical interest only, since it typically represents an over-fitted model. Thus, we would want to choose a good regularized model. Our results indicate that Boosting gives a natural method for doing that, by "stopping early" in the boosting process. Furthermore, they point out the fundamental similarity between Boosting and SVMs: both approaches allow us to fit regularized models in high-dimensional predictor space, using a computational trick. They differ in the regularization approach they take—exact l_2 regularization for SVMs, approximate l_1 regularization for Boosting—-and in the computational trick that facilitates fitting—the "kernel" trick for SVMs, coordinate descent for Boosting.

1.1 Related Work

Schapire et al. (1998) have identified the normalized margins as distance from an l_1-normed separating hyper-plane. Their results relate the boosting iterations' success to the minimal margin of the combined model. Rätsch et al. (2001b) take this further using an asymptotic analysis of AdaBoost. They prove that the "normalized" minimal margin, $\min_i y_i \sum_t \alpha_t h_t(\mathbf{x}_i) / \sum_t |\alpha_t|$, is asymptotically equal for both classes. In other words, they prove that the asymptotic separating hyper-plane is equally far away from the closest points on either side. This is a property of the margin maximizing separating hyper-plane as we define it. Both papers also illustrate the margin maximizing effects of AdaBoost through experimentation. However, they both stop short of proving the convergence to optimal (margin maximizing) solutions.

Motivated by our result, Rätsch and Warmuth (2002) have recently asserted the margin-maximizing properties of ε-AdaBoost, using a different approach than the one used in this paper. Their results relate only to the asymptotic convergence of infinitesimal AdaBoost, compared to our analysis of the "regularized path" traced along the way and of a variety of boosting loss functions, which also leads to a convergence result on binomial log-likelihood loss.

The convergence of boosting to an "optimal" solution from a loss function perspective has been analyzed in several papers. Rätsch et al. (2001a) and Collins et al. (2000) give results and bounds on the convergence of training-set loss, $\sum_i C(y_i, \sum_t \alpha_t h_t(\mathbf{x}_i))$, to its minimum. However, in the separable case convergence of the loss to 0 is inherently different from convergence of the linear separator to the optimal separator. Any solution which separates the two classes perfectly can drive the exponential (or log-likelihood) loss to 0, simply by scaling coefficients up linearly.

Two recent papers have made the connection between boosting and l_1 regularization in a slightly different context than this paper. Zhang (2003) suggests a "shrinkage" version of boosting which converges to l_1 regularized solutions, while Zhang and Yu (2003) illustrate the quantitative relationship between early stopping in boosting and l_1 constraints.

2. Boosting as Gradient Descent

Generic gradient-based boosting algorithms (Friedman, 2001; Mason et al., 1999) attempt to find a good linear combination of the members of some dictionary of basis functions to optimize a given loss function over a sample. This is done by searching, at each iteration, for the basis function which gives the "steepest descent" in the loss, and changing its coefficient accordingly. In other words, this is a "coordinate descent" algorithm in $\mathbb{R}^J$, where we assign one dimension (or coordinate) for the coefficient of each dictionary function.

Assume we have data $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$, a loss (or cost) function $C(y,F)$, and a set of dictionary functions $\{h_j(\mathbf{x})\} : \mathbb{R}^d \to \mathbb{R}$. Then all of these algorithms follow the same essential steps:

Algorithm 1 *Generic gradient-based boosting algorithm*

1. *Set* $\beta^{(0)} = 0$.
2. *For* $t = 1 : T$,
   (a) *Let* $F_i = \beta^{(t-1)'} h(\mathbf{x}_i),\ i = 1,\ldots,n$ *(the current fit).*
   (b) *Set* $w_i = \frac{\partial C(y_i,F_i)}{\partial F_i},\ i = 1,\ldots,n.$
   (c) *Identify* $j_t = \arg\max_j |\sum_i w_i h_j(\mathbf{x}_i)|$.
   (d) *Set* $\beta_{j_t}^{(t)} = \beta_{j_t}^{(t-1)} - \alpha_t sign(\sum_i w_i h_{j_t}(\mathbf{x}_i))$ *and* $\beta_k^{(t)} = \beta_k^{(t-1)}, k \neq j_t$.

Here $\beta^{(t)}$ is the "current" coefficient vector and $\alpha_t > 0$ is the current step size. Notice that $\sum_i w_i h_{j_t}(\mathbf{x}_i) = \frac{\partial \sum_i C(y_i,F_i)}{\partial \beta_{j_t}}$.

As we mentioned, Algorithm 1 can be interpreted simply as a coordinate descent algorithm in "weak learner" space. Implementation details include the dictionary $\mathcal{H}$ of "weak learners", the loss function $C(y,F)$, the method of searching for the optimal j_t and the way in which α_t is determined.[3] For example, the original AdaBoost algorithm uses this scheme with the exponential loss $C(y,F) = \exp(-yF)$, and an implicit line search to find the best α_t once a "direction" j_t has been chosen (see Hastie et al., 2001; Mason et al., 1999). The dictionary used by AdaBoost in this formulation would be a set of candidate classifiers, i.e., $h_j(\mathbf{x}_i) \in \{-1,+1\}$—usually decision trees are used in practice.

2.1 Practical Implementation of Boosting

The dictionaries used for boosting are typically very large—practically infinite—and therefore the generic boosting algorithm we have presented cannot be implemented verbatim. In particular, it is not practical to exhaustively search for the maximizer in step 2(c). Instead, an approximate, usually greedy search is conducted to find a "good" candidate weak learner h_{j_t} which makes the first order decline in the loss large (even if not maximal among all possible models).

In the common case that the dictionary of weak learners is comprised of decision trees with up to k nodes, the way AdaBoost and other boosting algorithms solve stage 2(c) is by building a

3. The sign of α_t will always be $-sign(\sum_i w_i h_{j_t}(\mathbf{x}_i))$, since we want the loss to be reduced. In most cases, the dictionary $\mathcal{H}$ is negation closed, and so it can be assumed WLOG that the coefficients are always positive and increasing

decision tree to a re-weighted version of the data, with the weights $|w_i|$. Thus they first replace step 2(c) with minimization of

$$\sum_i |w_i| 1\{y_i \neq h_{j_t}(\mathbf{x}_i)\},$$

which is easily shown to be equivalent to the original step 2(c). They then use a greedy decision-tree building algorithm such as CART or C5 to build a k-node decision tree which minimizes this quantity, i.e., achieves low "weighted misclassification error" on the weighted data. Since the tree is built greedily—one split at a time—it will not be the global minimizer of weighted misclassification error among all k-node decision trees. However, it will be a good fit for the re-weighted data, and can be considered an approximation to the optimal tree.

This use of approximate optimization techniques is critical, since much of the strength of the boosting approach comes from its ability to build additive models in *very* high-dimensional predictor spaces. In such spaces, standard exact optimization techniques are impractical: any approach which requires calculation and inversion of Hessian matrices is completely out of the question, and even approaches which require only first derivatives, such as coordinate descent, can only be implemented approximately.

2.2 Gradient-Based Boosting as a Generic Modeling Tool

As Friedman (2001); Mason et al. (1999) mention, this view of boosting as gradient descent allows us to devise boosting algorithms for any function estimation problem—all we need is an appropriate loss and an appropriate dictionary of "weak learners". For example, Friedman et al. (2000) suggested using the binomial log-likelihood loss instead of the exponential loss of AdaBoost for binary classification, resulting in the LogitBoost algorithm. However, there is no need to limit boosting algorithms to classification—Friedman (2001) applied this methodology to regression estimation, using squared error loss and regression trees, and Rosset and Segal (2003) applied it to density estimation, using the log-likelihood criterion and Bayesian networks as weak learners. Their experiments and those of others illustrate that the practical usefulness of this approach—coordinate descent in high dimensional predictor space—carries beyond classification, and even beyond supervised learning.

The view we present in this paper, of coordinate-descent boosting as approximate l_1-regularized fitting, offers some insight into why this approach would be good in general: it allows us to fit regularized models directly in high dimensional predictor space. In this it bears a conceptual similarity to support vector machines, which exactly fit an l_2 regularized model in high dimensional (RKH) predictor space.

2.3 Loss Functions

The two most commonly used loss functions for boosting classification models are the exponential and the (minus) binomial log-likelihood:

$$\begin{aligned} Exponential: &\quad C_e(y,F) = \exp(-yF); \\ Log likelihood: &\quad C_l(y,F) = \log(1+\exp(-yF)). \end{aligned}$$

These two loss functions bear some important similarities to each other. As Friedman et al. (2000) show, the population minimizer of expected loss at point $\mathbf{x}$ is similar for both loss functions and is

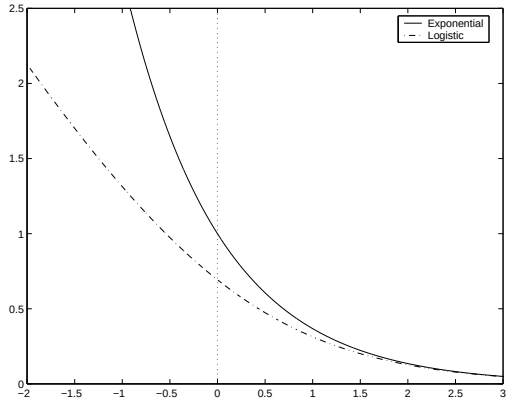


Figure 2: The two classification loss functions

given by

$$\hat{F}(\mathbf{x}) = c \cdot log\left[\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})}\right],$$

where $c_e = 1/2$ for exponential loss and $c_l = 1$ for binomial loss.

More importantly for our purpose, we have the following simple proposition, which illustrates the strong similarity between the two loss functions for positive margins (i.e., correct classifications):

Proposition 1

$$yF \geq 0 \Rightarrow 0.5C_e(y,F) \leq C_l(y,F) \leq C_e(y,F). \tag{5}$$

In other words, the two losses become similar if the margins are positive, and both behave like exponentials.

Proof Consider the functions $f_1(z) = z$ and $f_2(z) = log(1+z)$ for $z \in [0,1]$. Then $f_1(0) = f_2(0) = 0$, and

$$\frac{\partial f_1(z)}{\partial z} \equiv 1$$

$$\frac{1}{2} \leq \frac{\partial f_2(z)}{\partial z} = \frac{1}{1+z} \leq 1.$$

Thus we can conclude $0.5 f_1(z) \leq f_2(z) \leq f_1(z)$. Now set $z = exp(-yf)$ and we get the desired result. ■

For negative margins the behaviors of C_e and C_l are very different, as Friedman et al. (2000) have noted. In particular, C_l is more robust against outliers and misspecified data.

2.4 Line-Search Boosting vs. ε-Boosting

As mentioned above, AdaBoost determines α_t using a line search. In our notation for Algorithm 1 this would be

$$\alpha_t = \arg\min_{\alpha} \sum_i C(y_i, F_i + \alpha h_{j_t}(\mathbf{x_i})).$$

The alternative approach, suggested by Friedman (2001); Hastie et al. (2001), is to "shrink" all α_t to a single small value ε. This may slow down learning considerably (depending on how small ε is), but is attractive theoretically: the first-order theory underlying gradient boosting implies that the weak learner chosen is the best increment only "locally". It can also be argued that this approach is "stronger" than line search, as we can keep selecting the same h_{j_t} repeatedly if it remains optimal and so ε-boosting dominates line-search boosting in terms of training error. In practice, this approach of "slowing the learning rate" usually performs better than line-search in terms of prediction error as well (see Friedman, 2001). For our purposes, we will mostly assume ε is infinitesimally small, so the theoretical boosting algorithm which results is the "limit" of a series of boosting algorithms with shrinking ε.

In regression terminology, the line-search version is equivalent to forward stage-wise modeling, infamous in the statistics literature for being too greedy and highly unstable (see Friedman, 2001). This is intuitively obvious, since by increasing the coefficient until it saturates we are destroying "signal" which may help us select other good predictors.

3. l_p Margins, Support Vector Machines and Boosting

We now introduce the concept of margins as a geometric interpretation of a binary classification model. In the context of boosting, this view offers a different understanding of AdaBoost from the gradient descent view presented above. In the following sections we connect the two views.

3.1 The Euclidean Margin and the Support Vector Machine

Consider a classification model in high dimensional predictor space: $F(\mathbf{x}) = \sum_j h_j(\mathbf{x})\beta_j$. We say that the model *separates* the training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$ if $sign(F(\mathbf{x}_i)) = y_i, \ \forall i$. From a geometrical perspective this means that the hyper-plane defined by $F(\mathbf{x}) = 0$ is a separating hyper-plane for this data, and we define its (Euclidean) margin as

$$m_2(\beta) = \min_i \frac{y_i F(\mathbf{x}_i)}{\|\beta\|_2}. \tag{6}$$

The margin-maximizing separating hyper-plane for this data would be defined by β which maximizes $m_2(\beta)$. Figure 3 shows a simple example of separable data in two dimensions, with its margin-maximizing separating hyper-plane. The Euclidean margin-maximizing separating hyper-plane is the (non regularized) support vector machine solution. Its margin maximizing properties play a central role in deriving generalization error bounds for these models, and form the basis for a rich literature.

3.2 The l_1 Margin and Its Relation to Boosting

Instead of considering the Euclidean margin as in (6) we can define an "l_p margin" concept as

$$m_p(\beta) = \min_i \frac{y_i F(\mathbf{x}_i)}{\|\beta\|_p}. \tag{7}$$

Of particular interest to us is the case $p = 1$. Figure 4 shows the l_1 margin maximizing separating hyper-plane for the same simple example as Figure 3. Note the fundamental difference between

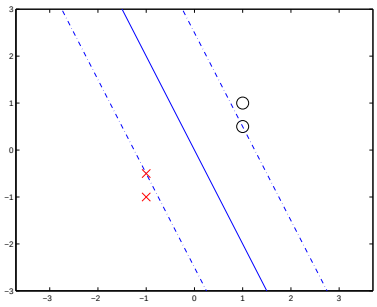


Figure 3: A simple data example, with two observations from class "O" and two observations from class "X". The full line is the Euclidean margin-maximizing separating hyper-plane.

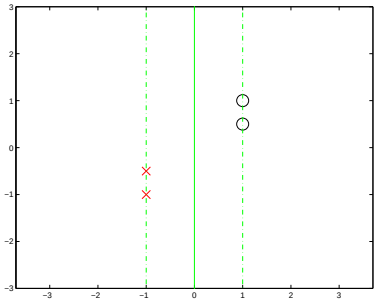


Figure 4: l_1 margin maximizing separating hyper-plane for the same data set as Figure 3. The difference between the diagonal Euclidean optimal separator and the vertical l_1 optimal separator illustrates the "sparsity" effect of optimal l_1 separation

the two solutions: the l_2-optimal separator is diagonal, while the l_1-optimal one is vertical. To understand why this is so we can relate the two margin definitions to each other as

$$\frac{yF(\mathbf{x})}{\|\beta\|_1} = \frac{yF(\mathbf{x})}{\|\beta\|_2} \cdot \frac{\|\beta\|_2}{\|\beta\|_1}. \tag{8}$$

From this representation we can observe that the l_1 margin will tend to be big if the ratio $\frac{\|\beta\|_2}{\|\beta\|_1}$ is big. This ratio will generally be big if β is sparse. To see this, consider fixing the l_1 norm of the vector and then comparing the l_2 norm of two candidates: one with many small components and the other—a sparse one—with a few large components and many zero components. It is easy to see that the second vector will have bigger l_2 norm, and hence (if the l_2 margin for both vectors is equal) a bigger l_1 margin.

A different perspective on the difference between the optimal solutions is given by a theorem due to Mangasarian (1999), which states that the l_p margin maximizing separating hyper plane maximizes the l_q distance from the closest points to the separating hyper-plane, with $\frac{1}{p} + \frac{1}{q} = 1$. Thus the Euclidean optimal separator ($p = 2$) also maximizes Euclidean distance between the points and the hyper-plane, while the l_1 optimal separator maximizes l_∞ distance. This interesting result gives another intuition why l_1 optimal separating hyper-planes tend to be coordinate-oriented (i.e., have sparse representations): since l_∞ projection considers only the largest coordinate distance, some coordinate distances may be 0 at no cost of decreased l_∞ distance.

Schapire et al. (1998) have pointed out the relation between AdaBoost and the l_1 margin. They prove that, in the case of separable data, the boosting iterations increase the "boosting" margin of the model, defined as

$$\min_i \frac{y_i F(\mathbf{x_i})}{\|\alpha\|_1}. \tag{9}$$

In other words, this is the l_1 margin of the model, except that it uses the α incremental representation rather than the β "geometric" representation for the model. The two representations give the same l_1 norm if there is sign consistency, or "monotonicity" in the coefficient paths traced by the model, i.e., if at every iteration t of the boosting algorithm

$$\beta_{j_t} \neq 0 \Rightarrow sign(\alpha_t) = sign(\beta_{j_t}). \tag{10}$$

As we will see later, this monotonicity condition will play an important role in the equivalence between boosting and l_1 regularization.

The l_1-margin maximization view of AdaBoost presented by Schapire et al. (1998)—and a whole plethora of papers that followed—is important for the analysis of boosting algorithms for two distinct reasons:

- It gives an intuitive, geometric interpretation of the model that AdaBoost is looking for—a model which separates the data well in this l_1-margin sense. Note that the view of boosting as gradient descent in a loss criterion doesn't really give the same kind of intuition: if the data is separable, then any model which separates the training data will drive the exponential or binomial loss to 0 when scaled up:

$$m_1(\beta) > 0 \implies \sum_i C(y_i, d\beta' \mathbf{x}_i) \to 0 \ \ as \ \ d \to \infty.$$

- The l_1-margin behavior of a classification model on its training data facilitates generation of generalization (or prediction) error bounds, similar to those that exist for support vector machines (Schapire et al., 1998). The important quantity in this context is not the margin but the "normalized" margin, which considers the "conjugate norm" of the predictor vectors:

$$\frac{y_i \beta' h(x_i)}{\|\beta\|_1 \|h(x_i)\|_\infty}.$$

  When the dictionary we are using is comprised of classifiers then $\|h(x_i)\|_\infty \equiv 1$ always and thus the l_1 margin is *exactly* the relevant quantity. The error bounds described by Schapire et al. (1998) allow using the whole l_1 margin distribution, not just the minimal margin. However, boosting's tendency to separate well in the l_1 sense is a central motivation behind their results.

From a statistical perspective, however, we should be suspicious of margin-maximization as a method for building good prediction models in high dimensional predictor space. Margin maximization in high dimensional space is likely to lead to over-fitting and bad prediction performance. This has been observed in practice by many authors, in particular Breiman (1999). Our results in the next two sections suggest an explanation based on model complexity: margin maximization is the limit of parametric regularized optimization models, as the regularization vanishes, and the regularized models along the path may well be superior to the margin maximizing "limiting" model, in terms of prediction performance. In Section 7 we return to discuss these issues in more detail.

4. Boosting as Approximate Incremental l_1 Constrained Fitting

In this section we introduce an interpretation of the generic coordinate-descent boosting algorithm as tracking a path of approximate solutions to l_1-constrained (or equivalently, regularized) versions of its loss criterion. This view serves our understanding of what boosting does, in particular the connection between early stopping in boosting and regularization. We will also use this view to get a result about the asymptotic margin-maximization of regularized classification models, and by analogy of classification boosting. We build on ideas first presented by Hastie et al. (2001, Chapter 10) and Efron et al. (2004).

Given a convex non-negative loss criterion $C(\cdot,\cdot)$, consider the 1-dimensional path of optimal solutions to l_1 constrained optimization problems over the training data:

$$\hat{\beta}(c) = \arg\min_{\|\beta\|_1 \le c} \sum_i C(y_i, h(\mathbf{x}_i)'\beta). \tag{11}$$

As c varies, we get that $\hat{\beta}(c)$ traces a 1-dimensional "optimal curve" through $\mathbb{R}^J$. If an optimal solution for the non-constrained problem exists and has finite l_1 norm c_0, then obviously $\hat{\beta}(c) = \hat{\beta}(c_0) = \hat{\beta}\ ,\ \forall c > c_0$. in the case of separable 2-class data, using either C_e or C_l, there is no finite-norm optimal solution. Rather, the constrained solution will always have $\|\hat{\beta}(c)\|_1 = c$.

A different way of building a solution which has l_1 norm c, is to run our ε-boosting algorithm for c/ε iterations. This will give an $\alpha^{(c/\varepsilon)}$ vector which has l_1 norm exactly c. For the norm of the geometric representation $\beta^{(c/\varepsilon)}$ to also be equal to c, we need the monotonicity condition (10) to hold as well. This condition will play a key role in our exposition.

We are going to argue that the two solution paths $\hat{\beta}(c)$ and $\beta^{(c/\varepsilon)}$ are very similar for ε "small". Let us start by observing this similarity in practice. Figure 1 in the introduction shows an example of

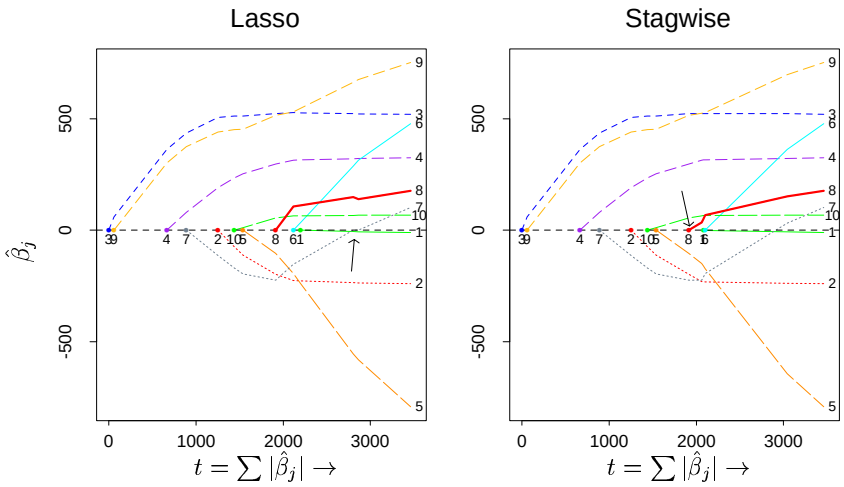


Figure 5: Another example of the equivalence between the Lasso optimal solution path (left) and ε-boosting with squared error loss. Note that the equivalence breaks down when the path of variable 7 becomes non-monotone

this similarity for squared error loss fitting with l_1 (lasso) penalty. Figure 5 shows another example in the same mold, taken from Efron et al. (2004). The data is a diabetes study and the "dictionary" used is just the original 10 variables. The panel on the left shows the path of optimal l_1-constrained solutions $\hat{\beta}(c)$ and the panel on the right shows the ε-boosting path with the 10-dimensional dictionary (the total number of boosting iterations is about 6000). The 1-dimensional path through $\mathbb{R}^{10}$ is described by 10 coordinate curves, corresponding to each one of the variables. The interesting phenomenon we observe is that the two coefficient traces are not completely identical. Rather, they agree up to the point where variable 7 coefficient path becomes *non monotone*, i.e., it violates (10) (this point is where variable 8 comes into the model, see the arrow on the right panel). This example illustrates that the monotonicity condition—and its implication that $\|\alpha\|_1 = \|\beta\|_1$—is critical for the equivalence between ε-boosting and l_1-constrained optimization.

The two examples we have seen so far have used squared error loss, and we should ask ourselves whether this equivalence stretches beyond this loss. Figure 6 shows a similar result, but this time for the binomial log-likelihood loss, C_l. We used the "spam" data set, taken from the UCI repository (Blake and Merz, 1998). We chose only 5 predictors of the 57 to make the plots more interpretable and the computations more accommodating. We see that there is a perfect equivalence between the exact constrained solution (i.e., regularized logistic regression) and ε-boosting in this case, since the paths are fully monotone.

To justify why this observed equivalence is not surprising, let us consider the following "l_1-locally optimal monotone direction" problem of finding the best monotone ε increment to a given model β_0:

$$\begin{aligned} \min \quad & C(\beta) \\ s.t. \quad & \|\beta\|_1 - \|\beta_0\|_1 \leq \varepsilon, \\ & |\beta| \succeq |\beta_0| \text{ (component-wise).} \end{aligned} \tag{12}$$

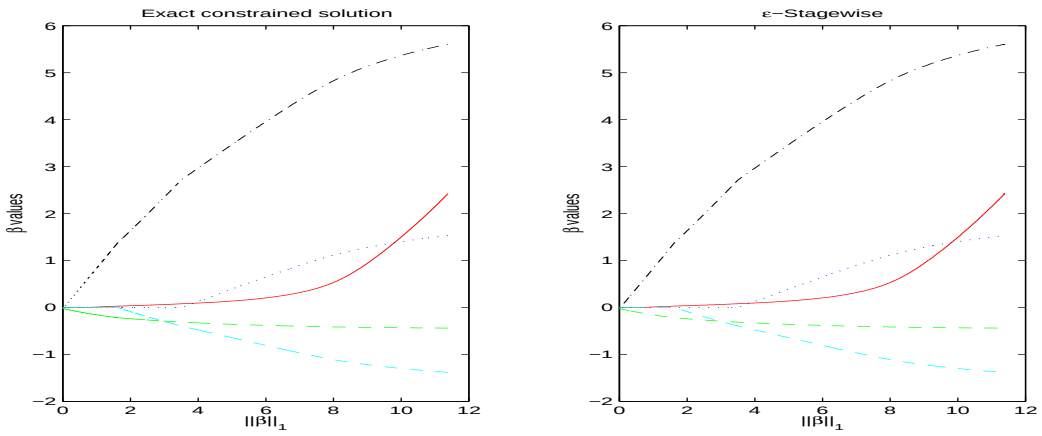


Figure 6: Exact coefficient paths (left) for l_1-constrained logistic regression and boosting coefficient paths (right) with binomial log-likelihood loss on five variables from the "spam" data set. The boosting path was generated using $\varepsilon = 0.003$ and 7000 iterations.

Here we use $C(\beta)$ as shorthand for $\sum_i C(y_i, h(\mathbf{x}_i)'\beta)$. A first order Taylor expansion gives us

$$C(\beta) = C(\beta_0) + \nabla C(\beta_0)'(\beta - \beta_0) + O(\varepsilon^2).$$

And given the l_1 constraint on the increase in $\|\beta\|_1$, it is easy to see that a first-order optimal solution (and therefore an optimal solution as $\varepsilon \to 0$) will make a "coordinate descent" step, i.e.

$$\beta_j \neq \beta_{0,j} \quad \Rightarrow \quad |\nabla C(\beta_0)_j| = \max_k |\nabla C(\beta_0)_k|,$$

assuming the signs match, i.e., $sign(\beta_{0j}) = -sign(\nabla C(\beta_0)_j)$.

So we get that if the optimal solution to (12) *without* the monotonicity constraint happens to be monotone, then it is equivalent to a coordinate descent step. And so it is reasonable to expect that if the optimal l_1 regularized path is monotone (as it indeed is in Figures 1,6), then an "infinitesimal" ε-boosting algorithm would follow the same path of solutions. Furthermore, even if the optimal path is not monotone, we can still use the formulation (12) to argue that ε-boosting would tend to follow an approximate l_1-regularized path. The main difference between the ε-boosting path and the true optimal path is that it will tend to "delay" becoming non-monotone, as we observe for variable 7 in Figure 5. To understand this specific phenomenon would require analysis of the true optimal path, which falls outside the scope of our discussion—Efron et al. (2004) cover the subject for squared error loss, and their discussion applies to any continuously differentiable convex loss, using second-order approximations.

We can employ this understanding of the relationship between boosting and l_1 regularization to construct l_p boosting algorithms by changing the coordinate-selection criterion in the coordinate descent algorithm. We will get back to this point in Section 7, where we design an "l_2 boosting" algorithm.

The experimental evidence and heuristic discussion we have presented lead us to the following conjecture which connects slow boosting and l_1-regularized optimization:

Conjecture 2 *Consider applying the ε-boosting algorithm to any convex loss function, generating a path of solutions $\beta^{(\varepsilon)}(t)$. Then if the optimal coefficient paths are monotone $\forall c < c_0$, i.e., if $\forall j,\ |\hat{\beta}(c)_j|$ is non-decreasing in the range $c < c_0$, then*

$$\lim_{\varepsilon \to 0} \beta^{(\varepsilon)}(c_0/\varepsilon) = \hat{\beta}(c_0).$$

Efron et al. (2004, Theorem 2) prove a weaker "local" result for the case of squared error loss only. We generalize their result to any convex loss. However this result still does not prove the "global" convergence which the conjecture claims, and the empirical evidence implies. For the sake of brevity and readability, we defer this proof, together with concise mathematical definition of the different types of convergence, to appendix A.

In the context of "real-life" boosting, where the number of basis functions is usually very large, and making ε small enough for the theory to apply would require running the algorithm forever, these results should not be considered directly applicable. Instead, they should be taken as an intuitive indication that boosting—especially the ε version—is, indeed, approximating optimal solutions to the constrained problems it encounters along the way.

5. l_p-Constrained Classification Loss Functions

Having established the relation between boosting and l_1 regularization, we are going to turn our attention to the regularized optimization problem. By analogy, our results will apply to boosting as well. We concentrate on C_e and C_l, the two classification losses defined above, and the solution paths of their l_p constrained versions:

$$\hat{\beta}^{(p)}(c) = \arg \min_{\|\beta\|_p \le c} \sum_i C(y_i, \beta' h(\mathbf{x_i})). \tag{13}$$

where C is either C_e or C_l. As we discussed below Equation (11), if the training data is separable in $span(\mathcal{H})$, then we have $\|\hat{\beta}^{(p)}(c)\|_p = c$ for *all* values of c. Consequently

$$\|\frac{\hat{\beta}^{(p)}(c)}{c}\|_p = 1.$$

We may ask what are the convergence points of this sequence as $c \to \infty$. The following theorem shows that these convergence points describe "l_p-margin maximizing" separating hyper-planes.

Theorem 3 *Assume the data is separable, i.e., $\exists \beta\ s.t. \forall i,\ y_i \beta' h(\mathbf{x}_i) > 0$.*
Then for both C_e and C_l, every *convergence point of $\frac{\hat{\beta}(c)}{c}$ corresponds to an l_p-margin-maximizing separating hyper-plane.*
If the l_p-margin-maximizing separating hyper-plane is unique, then it is the unique convergence points, i.e.

$$\hat{\beta}^{(p)} = \lim_{c \to \infty} \frac{\hat{\beta}^{(p)}(c)}{c} = \arg \max_{\|\beta\|_p = 1} \min_i y_i \beta' h(\mathbf{x}_i). \tag{14}$$

Proof This proof applies to both C_e and C_l, given the property in (5). Consider two separating candidates β_1 and β_2 such that $\|\beta_1\|_p = \|\beta_2\|_p = 1$. Assume that β_1 separates better, i.e.

$$m_1 := \min_i y_i \beta_1' h(\mathbf{x}_i) > m_2 := \min_i y_i \beta_2' h(\mathbf{x}_i) > 0.$$

Then we have the following simple lemma:

Lemma 4 *There exists some $D = D(m_1, m_2)$ such that $\forall d > D$, $d\beta_1$ incurs smaller loss than $d\beta_2$, in other words:*

$$\sum_i C(y_i, d\beta_1' h(\mathbf{x}_i)) < \sum_i C(y_i, d\beta_2' h(\mathbf{x}_i)).$$

Given this lemma, we can now prove that any convergence point of $\frac{\hat{\beta}^{(p)}(c)}{c}$ must be an l_p-margin maximizing separator. Assume β^* is a convergence point of $\frac{\hat{\beta}^{(p)}(c)}{c}$. Denote its minimal margin on the data by m^*. If the data is separable, clearly $m^* > 0$ (since otherwise the loss of $d\beta^*$ does not even converge to 0 as $d \to \infty$).

Now, assume some $\tilde{\beta}$ with $\|\tilde{\beta}\|_p = 1$ has bigger minimal margin $\tilde{m} > m^*$. By continuity of the minimal margin in β, there exists some open neighborhood of β^*

$$N_{\beta^*} = \{\beta : \|\beta - \beta^*\|_2 < \delta\}$$

and an $\varepsilon > 0$, such that

$$\min_i y_i \beta' h(\mathbf{x_i}) < \tilde{m} - \varepsilon, \ \ \forall \beta \in N_{\beta^*}.$$

Now by the lemma we get that there exists some $D = D(\tilde{m}, \tilde{m} - \varepsilon)$ such that $d\tilde{\beta}$ incurs smaller loss than $d\beta$ for any $d > D$, $\beta \in N_{\beta^*}$. Therefore β^* *cannot be a convergence point of* $\frac{\hat{\beta}^{(p)}(c)}{c}$.

We conclude that any convergence point of the sequence $\frac{\hat{\beta}^{(p)}(c)}{c}$ must be an l_p-margin maximizing separator. If the margin maximizing separator is unique then it is the only possible convergence point, and therefore

$$\hat{\beta}^{(p)} = \lim_{c \to \infty} \frac{\hat{\beta}^{(p)}(c)}{c} = \arg \max_{\|\beta\|_p = 1} \min_i y_i \beta' h(\mathbf{x}_i).$$

■

Proof of Lemma Using (5) and the definition of C_e, we get for both loss functions:

$$\sum_i C(y_i, d\beta_1' h(\mathbf{x_i})) \leq n \exp(-d \cdot m_1).$$

Now, since β_1 separates better, we can find our desired

$$D = D(m_1, m_2) = \frac{logn + log2}{m_1 - m_2}$$

such that

$$\forall d > D, \ \ n \exp(-d \cdot m_1) < 0.5 \exp(-d \cdot m_2).$$

And using (5) and the definition of C_e again we can write

$$0.5 \exp(-d \cdot m_2) \leq \sum_i C(y_i, d\beta_2' h(\mathbf{x_i})).$$

Combining these three inequalities we get our desired result:

$$\forall d > D, \ \ \sum_i C(y_i, d\beta_1' h(\mathbf{x_i})) \leq \sum_i C(y_i, d\beta_2' h(\mathbf{x_i})).$$

■

We thus conclude that if the l_p-margin maximizing separating hyper-plane is unique, the normalized constrained solution converges to it. In the case that the margin maximizing separating hyper-plane is not unique, we can in fact prove a stronger result, which indicates that the limit of the regularized solutions would then be determined by the second smallest margin, then by the third and so on. This result is mainly of technical interest and we prove it in Appendix B, Section 2.

5.1 Implications of Theorem 3

We now briefly discuss the implications of this theorem for boosting and logistic regression.

5.1.1 BOOSTING IMPLICATIONS

Combined with our results from Section 4, Theorem 3 indicates that the normalized boosting path $\frac{\beta^{(t)}}{\sum_{u\le t}\alpha_u}$—with either C_e or C_l used as loss—"approximately" converges to a separating hyper-plane $\hat{\beta}$, which attains

$$\max_{\|\beta\|_1=1} \min_i y_i \beta' h(\mathbf{x_i}) = \max_{\|\beta\|_1=1} \|\beta\|_2 \min_i y_i d_i, \tag{15}$$

where d_i is the (signed) Euclidean distance from the training point i to the separating hyper-plane. In other words, it maximizes Euclidean distance scaled by an l_2 norm. As we have mentioned already, this implies that the *asymptotic* boosting solution will tend to be sparse in representation, due to the fact that for fixed l_1 norm, the l_2 norm of vectors that have many 0 entries will generally be larger. In fact, under rather mild conditions, the asymptotic solution $\hat{\beta} = \lim_{c\to\infty} \hat{\beta}^{(1)}(c)/c$, will have *at most n* (the number of observations) non-zero coefficients, if we use either C_l or C_e as the loss. See Appendix B, Section 1 for proof.

5.1.2 LOGISTIC REGRESSION IMPLICATIONS

Recall, that the logistic regression (maximum likelihood) solution is undefined if the data is separable in the Euclidean space spanned by the predictors. Theorem 3 allows us to define a logistic regression solution for separable data, as follows:

1. Set a high constraint value c_{max}
2. Find $\hat{\beta}^{(p)}(c_{max})$, the solution to the logistic regression problem subject to the constraint $\|\beta\|_p \le c_{max}$. The problem is convex for any $p \ge 1$ and differentiable for any $p > 1$, so interior point methods can be used to solve this problem.
3. Now you have (approximately) the l_p-margin maximizing solution for this data, described by

$$\frac{\hat{\beta}^{(p)}(c_{max})}{c_{max}}.$$

   This is a solution to the original problem in the sense that it is, approximately, the convergence point of the normalized l_p-constrained solutions, as the constraint is relaxed.

Of course, with our result from Theorem 3 it would probably make more sense to simply find the optimal separating hyper-plane directly—this is a linear programming problem for l_1 separation and a quadratic programming problem for l_2 separation. We can then consider this optimal separator as a logistic regression solution for the separable data.

6. Examples

We now apply boosting to several data sets and interpret the results in light of our regularization and margin-maximization view.

6.1 Spam Data Set

We now know if the data are separable and we let boosting run forever, we will approach the same "optimal" separator for both C_e and C_l. However if we stop early—or if the data is not separable—the behavior of the two loss functions may differ significantly, since C_e weighs negative margins exponentially, while C_l is approximately linear in the margin for large negative margins (see Friedman et al., 2000). Consequently, we can expect C_e to concentrate more on the "hard" training data, in particular in the non-separable case. Figure 7 illustrates the behavior of ε-boosting with both

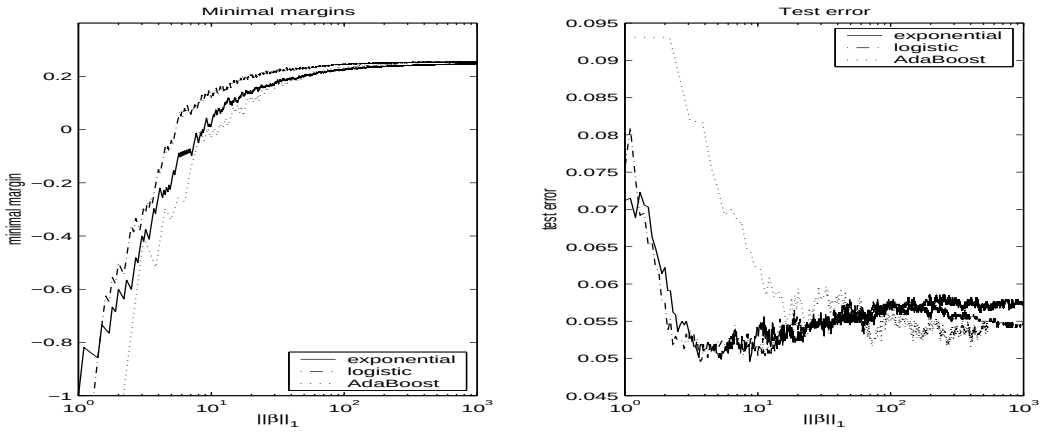


Figure 7: Behavior of boosting with the two loss functions on spam data set

loss functions, as well as that of AdaBoost, on the spam data set (57 predictors, binary response). We used 10 node trees and $\varepsilon = 0.1$. The left plot shows the minimal margin as a function of the l_1 norm of the coefficient vector $\|\beta\|_1$. Binomial loss creates a bigger minimal margin initially, but the minimal margins for both loss functions are converging asymptotically. AdaBoost initially lags behind but catches up nicely and reaches the same minimal margin asymptotically. The right plot shows the test error as the iterations proceed, illustrating that both ε-methods indeed seem to over-fit eventually, even as their "separation" (minimal margin) is still improving. AdaBoost did not significantly over-fit in the 1000 iterations it was allowed to run, but it obviously would have if it were allowed to run on.

We should emphasize that the comparison between AdaBoost and ε-boosting presented considers as a basis for comparison the l_1 norm, not the number of iterations. In terms of computational complexity, as represented by the number of iterations, AdaBoost reaches both a large minimal mar-

gin and good prediction performance much more quickly than the "slow boosting" approaches, as AdaBoost tends to take larger steps.

6.2 Simulated Data

To make a more educated comparison and more compelling visualization, we have constructed an example of separation of 2-dimensional data using a 8-th degree polynomial dictionary (45 functions). The data consists of 50 observations of each class, drawn from a mixture of Gaussians, and presented in Figure 8. Also presented, in the solid line, is the optimal l_1 separator for this data in this dictionary (easily calculated as a linear programming problem - note the difference from the l_2 optimal decision boundary, presented in Section 7.1, Figure 11). The optimal l_1 separator has only 12 non-zero coefficients out of 45.

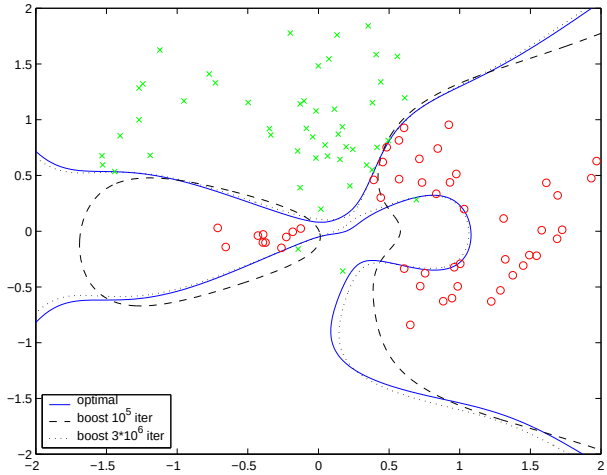


Figure 8: Artificial data set with l_1-margin maximizing separator (solid), and boosting models after 10^5 iterations (dashed) and 10^6 iterations (dotted) using $\varepsilon = 0.001$. We observe the convergence of the boosting separator to the optimal separator

We ran an ε-boosting algorithm on this data set, using the logistic log-likelihood loss C_l, with $\varepsilon = 0.001$, and Figure 8 shows two of the models generated after 10^5 and $3 \cdot 10^6$ iterations. We see that the models seem to converge to the optimal separator. A different view of this convergence is given in Figure 9, where we see two measures of convergence: the minimal margin (left, maximum value obtainable is the horizontal line) and the l_1-norm distance between the normalized models (right), given by

$$\sum_j \left| \hat{\beta}_j - \frac{\beta_j^{(t)}}{\|\beta^{(t)}\|_1} \right|,$$

where $\hat{\beta}$ is the optimal separator with l_1 norm 1 and $\beta^{(t)}$ is the boosting model after t iterations.

We can conclude that on this simple artificial example we get nice convergence of the logistic-boosting model path to the l_1-margin maximizing separating hyper-plane.

We can also use this example to illustrate the similarity between the boosted path and the path of l_1 optimal solutions, as we have discussed in Section 4.

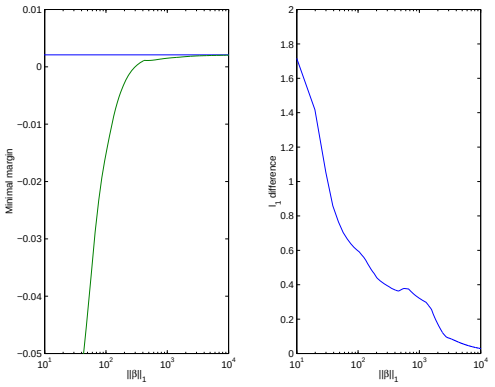


Figure 9: Two measures of convergence of boosting model path to optimal l_1 separator: minimal margin (left) and l_1 distance between the normalized boosting coefficient vector and the optimal model (right)

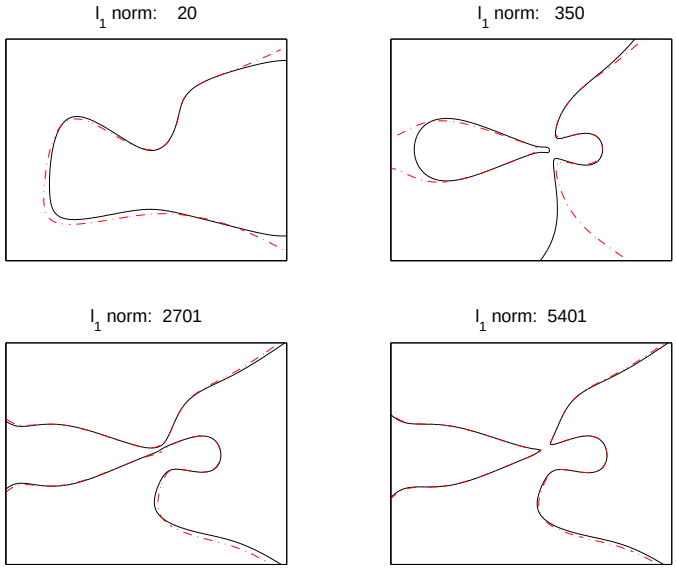


Figure 10: Comparison of decision boundary of boosting models (broken) and of optimal constrained solutions with same norm (full)

Figure 10 shows the class decision boundaries for 4 models generated along the boosting path, compared to the optimal solutions to the constrained "logistic regression" problem with the same bound on the l_1 norm of the coefficient vector. We observe the clear similarities in the way the solutions evolve and converge to the optimal l_1 separator. The fact that they differ (in some cases significantly) is not surprising if we recall the monotonicity condition presented in Section 4 for exact correspondence between the two model paths. In this case if we look at the coefficient paths

(not shown), we observe that the monotonicity condition is consistently violated in the low norm ranges, and hence we can expect the paths to be similar in spirit but not identical.

7. Discussion

We can now summarize what we have learned about boosting from the previous sections:

- Boosting approximately follows the path of l_1-regularized models for its loss criterion
- If the loss criterion is the exponential loss of AdaBoost or the binomial log-likelihood loss of logistic regression, then the l_1 regularized model converges to an l_1-margin maximizing separating hyper-plane, if the data are separable in the span of the weak learners

We may ask, which of these two points is the key to the success of boosting approaches. One empirical clue to answering this question, can be found in Breiman (1999), who programmed an algorithm to *directly* maximize the margins. His results were that his algorithm consistently got significantly higher minimal margins than AdaBoost on many data sets (and, in fact, a "higher" margin distribution beyond the minimal margin), but had slightly worse prediction performance. His conclusion was that margin maximization is *not* the key to AdaBoost's success. From a statistical perspective we can embrace this conclusion, as reflecting the importance of regularization in high-dimensional predictor space. By our results from the previous sections, "margin maximization" can be viewed as the limit of parametric regularized models, as the regularization vanishes.[4] Thus we would generally expect the margin maximizing solutions to perform *worse* than regularized models. In the case of boosting, regularization would correspond to "early stopping" of the boosting algorithm.

7.1 Boosting and SVMs as Regularized Optimization in High-dimensional Predictor Spaces

Our exposition has led us to view boosting as an approximate way to solve the regularized optimization problem

$$\min_{\beta} \sum_i C(y_i, \beta' h(\mathbf{x}_i)) + \lambda \|\beta\|_1 \tag{16}$$

which converges as $\lambda \to 0$ to $\hat{\beta}^{(1)}$, if our loss is C_e or C_l. In general, the loss C can be any convex differentiable loss and should be defined to match the problem domain.

Support vector machines can be described as solving the regularized optimization problem (see Friedman et al., 2000, Chapter 12)

$$\min_{\beta} \sum_i (1 - y_i \beta' h(\mathbf{x}_i))_+ + \lambda \|\beta\|_2^2 \tag{17}$$

which "converges" as $\lambda \to 0$ to the non-regularized support vector machine solution, i.e., the optimal Euclidean separator, which we denoted by $\hat{\beta}^{(2)}$.

An interesting connection exists between these two approaches, in that they allow us to solve the regularized optimization problem in high dimensional predictor space:

4. It can be argued that margin-maximizing models are still "regularized" in some sense, as they minimize a norm criterion among all separating models. This is arguably the property which still allows them to generalize reasonably well in many cases.

- We are able to solve the l_1- regularized problem approximately in very high dimension via boosting by applying the "approximate coordinate descent" trick of building a decision tree (or otherwise greedily selecting a weak learner) based on re-weighted versions of the data.

- Support vector machines facilitate a different trick for solving the regularized optimization problem in high dimensional predictor space: the "kernel trick". If our dictionary $\mathcal{H}$ spans a Reproducing Kernel Hilbert Space, then RKHS theory tells us we can find the regularized solutions by solving an n-dimensional problem, in the space spanned by the kernel representers $\{K(\mathbf{x}_i, \mathbf{x})\}$. This fact is by no means limited to the hinge loss of (17), and applies to any convex loss. We concentrate our discussion on SVM (and hence hinge loss) only since it is by far the most common and well-known application of this result.

So we can view both boosting and SVM as methods that allow us to fit regularized models in high dimensional predictor space using a computational "shortcut". The complexity of the model built is controlled by regularization. These methods are distinctly different than traditional statistical approaches for building models in high dimension, which start by reducing the dimensionality of the problem so that standard tools (e.g., Newton's method) can be applied to it, and also to make over-fitting less of a concern. While the merits of regularization without dimensionality reduction—like Ridge regression or the Lasso—are well documented in statistics, computational issues make it impractical for the size of problems typically solved via boosting or SVM, without computational tricks.

We believe that this difference may be a significant reason for the enduring success of boosting and SVM in data modeling, i.e.:

> Working in high dimension and regularizing is statistically preferable to a two-step procedure of first reducing the dimension, then fitting a model in the reduced space.

It is also interesting to consider the differences between the two approaches, in the loss (flexible vs. hinge loss), the penalty (l_1 vs. l_2), and the type of dictionary used (usually trees vs. RKHS). These differences indicate that the two approaches will be useful for different situations. For example, if the true model has a sparse representation in the chosen dictionary, then l_1 regularization may be warranted; if the form of the true model facilitates description of the class probabilities via a logistic-linear model, then the logistic loss C_l is the best loss to use, and so on.

The computational tricks for both SVM and boosting limit the kind of regularization that can be used for fitting in high dimensional space. However, the problems can still be formulated and solved for different regularization approaches, as long as the dimensionality is low enough:

- Support vector machines can be fitted with an l_1 penalty, by solving the *1-norm* version of the SVM problem, equivalent to replacing the l_2 penalty in (17) with an l_1 penalty. In fact, the 1-norm SVM is used quite widely, because it is more easily solved in the "linear", non-RKHS, situation (as a linear program, compared to the standard SVM which is a quadratic program) and tends to give sparser solutions in the primal domain.

- Similarly, we describe below an approach for developing a "boosting" algorithm for fitting approximate l_2 regularized models.

Both of these methods are interesting and potentially useful. However they lack what is arguably the most attractive property of the "standard" boosting and SVM algorithms: a computational trick to allow fitting in high dimensions.

7.1.1 AN l_2 BOOSTING ALGORITHM

We can use our understanding of the relation of boosting to regularization and Theorem 3 to formulate l_p-boosting algorithms, which will approximately follow the path of l_p-regularized solutions and converge to the corresponding l_p-margin maximizing separating hyper-planes. Of particular interest is the l_2 case, since Theorem 3 implies that l_2-constrained fitting using C_l or C_e will build a regularized path to the optimal separating hyper-plane in the Euclidean (or SVM) sense.

To construct an l_2 boosting algorithm, consider the "equivalent" optimization problem (12), and change the step-size constraint to an l_2 constraint:

$$\|\beta\|_2 - \|\beta_0\|_2 \leq \varepsilon.$$

It is easy to see that the first order solution to this problem entails selecting for modification the coordinate which maximizes

$$\frac{\nabla C(\beta_0)_k}{\beta_{0,k}}$$

and that subject to monotonicity, this will lead to a correspondence to the locally l_2-optimal direction.

Following this intuition, we can construct an l_2 boosting algorithm by changing only step $2(c)$ of our generic boosting algorithm of Section 2 to

2(c)* Identify j_t which maximizes $\frac{|\sum_i w_i h_{j_t}(\mathbf{x}_i)|}{|\beta_{j_t}|}$.

Note that the need to consider the current coefficient (in the denominator) makes the l_2 algorithm appropriate for toy examples only. In situations where the dictionary of weak learner is prohibitively large, we will need to figure out a trick like the one we presented in Section 2.1, to allow us to make an approximate search for the optimizer of step 2(c)*.

Another problem in applying this algorithm to large problems is that we never choose the same dictionary function twice, until all have non-0 coefficients. This is due to the use of the l_2 penalty, where the current coefficient value affects the rate at which the penalty term is increasing. In particular, if $\beta_j = 0$ then increasing it causes the penalty term $\|\beta\|_2$ to increase at rate 0, to first order (which is all the algorithm is considering).

The convergence of our l_2 boosting algorithm on the artificial data set of Section 6.2 is illustrated in Figure 11. We observe that the l_2 boosting models do indeed approach the optimal l_2 separator. It is interesting to note the significant difference between the optimal l_2 separator as presented in Figure 11 and the optimal l_1 separator presented in Section 6.2 (Figure 8).

8. Summary and Future Work

In this paper we have introduced a new view of boosting in general, and two-class boosting in particular, comprised of two main points:

- We have generalized results from Efron et al. (2004) and Hastie et al. (2001), to describe boosting as approximate l_1-regularized optimization.
- We have shown that the exact l_1-regularized solutions converge to an l_1-margin maximizing separating hyper-plane.

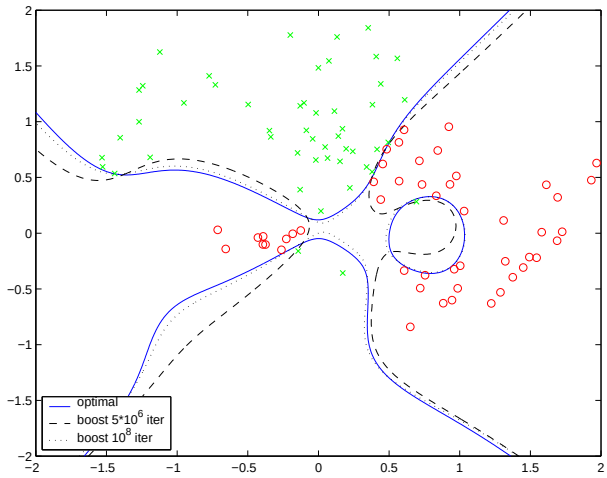


Figure 11: Artificial data set with l_2-margin maximizing separator (solid), and l_2-boosting models after $5*10^6$ iterations (dashed) and 10^8 iterations (dotted) using $\varepsilon = 0.0001$. We observe the convergence of the boosting separator to the optimal separator

We hope our results will help in better understanding how and why boosting works. It is an interesting and challenging task to separate the effects of the different components of a boosting algorithm:

- Loss criterion
- Dictionary and greedy learning method
- Line search / slow learning

and relate them to its success in different scenarios. The implicit l_1 regularization in boosting may also contribute to its success, as it has been shown that in some situations l_1 regularization is inherently superior to others (see Donoho et al., 1995).

An important issue when analyzing boosting is over-fitting in the noisy data case. To deal with over-fitting, Rätsch et al. (2001b) propose several regularization methods and generalizations of the original AdaBoost algorithm to achieve a *soft margin* by introducing slack variables. Our results indicate that the models along the boosting path can be regarded as l_1 regularized versions of the optimal separator, hence regularization can be done more directly and naturally by stopping the boosting iterations early. It is essentially a choice of the l_1 constraint parameter c.

Many other questions arise from our view of boosting. Among the issues to be considered:

- Is there a similar "separator" view of multi-class boosting? We have some tentative results to indicate that this might be the case if the boosting problem is formulated properly.
- Can the constrained optimization view of boosting help in producing generalization error bounds for boosting that would be more tight than the current existing ones?

Acknowledgments

We thank Stephen Boyd, Brad Efron, Jerry Friedman, Robert Schapire and Rob Tibshirani for helpful discussions. We thank the referees for their thoughtful and useful comments. This work was partially supported by Stanford graduate fellowship, grant DMS-0204612 from the National Science Foundation, and grant ROI-CA-72028-01 from the National Institutes of Health.

Appendix A. Local Equivalence of Infinitesimal ε-Boosting and l_1-Constrained Optimization

As before, we assume we have a set of training data $(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\dots(\mathbf{x}_n,y_n)$, a smooth cost function $C(y,F)$, and a set of basis functions $(h_1(\mathbf{x}),h_2(\mathbf{x}),\dots h_J(\mathbf{x}))$.

We denote by $\hat{\beta}(s)$ be the optimal solution of the l_1-constrained optimization problem:

$$\min_{\beta} \quad \sum_{i=1}^{n} C(y_i,h(\mathbf{x}_i)'\beta) \tag{18}$$

$$\text{subject to} \quad \|\beta\|_1 \le s. \tag{19}$$

Suppose we initialize the ε-boosting version of Algorithm 1, as described in Section 2, at $\hat{\beta}(s)$ and run the algorithm for T steps. Let $\beta(T)$ denote the coefficients after T steps.

The "global convergence" Conjecture 2 in Section 4 implies that $\forall \Delta s > 0$:

$$\beta(\Delta s/\varepsilon) \to \hat{\beta}(s+\Delta s) \text{ as } \varepsilon \to 0$$

under some mild assumptions. Instead of proving this "global" result, we show here a "local" result by looking at the derivative of $\hat{\beta}(s)$. Our proof builds on the proof by Efron et al. (2004, Theorem 2) of a similar result for the case that the cost is squared error loss $C(y,F)=(y-F)^2$. Theorem 1 below shows that if we start the ε-boosting algorithm at a solution $\hat{\beta}(s)$ of the l_1-constrained optimization problem (18)–(19), the "direction of change" of the ε-boosting solution will agree with that of the l_1-constrained optimization problem.

Theorem 1 *Assume the optimal coefficient paths $\hat{\beta}_j(s)\ \forall j$ are monotone in s and the coefficient paths $\beta_j(T)\ \forall j$ are also monotone as ε-boosting proceeds, then*

$$\frac{\beta(T)-\hat{\beta}(s)}{T\cdot\varepsilon} \to \nabla\hat{\beta}(s) \text{ as } \varepsilon \to 0, T \to \infty, T\cdot\varepsilon \to 0.$$

Proof First we introduce some notations. Let

$$\mathbf{h}_j = (h_j(\mathbf{x}_1),\dots h_j(\mathbf{x}_n))'$$

be the jth basis function evaluated at the n training data.

Let

$$\mathbf{F} = (F(\mathbf{x}_1),\dots F(\mathbf{x}_n))'$$

be the vector of current fit.

Let

$$\mathbf{r} = \left(-\frac{\partial C(y_1,F_1)}{\partial F_1},\dots -\frac{\partial C(y_n,F_n)}{\partial F_n}\right)'$$

be the current "generalized residual" vector as defined in Friedman (2001).

Let

$$c_j = \mathbf{h}_j'\mathbf{r}, \quad j = 1, \ldots J$$

be the current "correlation" between $\mathbf{h}_j$ and $\mathbf{r}$.

Let

$$\mathcal{A} = \{ j : |c_j| = \max_j |c_j| \}$$

be the set of indices for the maximum absolute correlation.

For clarity, we re-write this ε-boosting algorithm, starting from $\hat{\beta}(s)$, as a special case of Algorithm 1, as follows:

(1) Initialize $\beta(0) = \hat{\beta}(s), \mathbf{F}_0 = \mathbf{F}, \mathbf{r}_0 = \mathbf{r}$.

(2) For $t = 1 : T$

(a) Find $j_t = \arg\max_j |\mathbf{h}_j'\mathbf{r}_{t-1}|$.

(b) Update

$$\beta_{t,j_t} \leftarrow \beta_{t-1,j_t} + \varepsilon \cdot sign(c_{j_t})$$

(c) Update $\mathbf{F}_t$ and $\mathbf{r}_t$.

Notice in the above algorithm, we start from $\hat{\beta}(s)$, rather than 0. As proposed in Efron et al. (2004), we consider an idealized ε-boosting case: $\varepsilon \to 0$. As $\varepsilon \to 0$, $T \to \infty$ and $T \cdot \varepsilon \to 0$, under the monotone paths condition, Section 3.2 and Section 6 of Efron et al. (2004) showed

$$\frac{\mathbf{F}_T - \mathbf{F}_0}{T \cdot \varepsilon} \quad \to \quad \mathbf{u}, \tag{20}$$

$$\frac{\mathbf{r}_T - \mathbf{r}_0}{T \cdot \varepsilon} \quad \to \quad \mathbf{v}, \tag{21}$$

where $\mathbf{u}$ and $\mathbf{v}$ satisfy two constraints:

(**Constraint 1**) $\mathbf{u}$ is in the convex cone generated by $\{sign(c_j)\mathbf{h}_j : j \in \mathcal{A}\}$, i.e.:

$$\mathbf{u} = \sum_{j \in \mathcal{A}} P_j sign(c_j)\mathbf{h}_j, P_j \geq 0.$$

(**Constraint 2**) $\mathbf{v}$ has equal "correlation" with $sign(c_j)\mathbf{h}_j, j \in \mathcal{A}$:

$$sign(c_j)\mathbf{h}_j'\mathbf{v} = \lambda_{\mathcal{A}} \text{ for } j \in \mathcal{A}.$$

The first constraint is true because the basis functions in $\mathcal{A}^C$ will not be able to catch up in terms of $|c_j|$ for sufficiently small $T \cdot \varepsilon$; the P_j's are non-negative because the coefficient paths $\beta_j(T)$ are monotone. The second constraint can be seen by taking a Taylor expansion of $C(y,F)$ around F_0 to the quadratic term, letting $T \cdot \varepsilon$ go to zero and applying the result for the squared error loss from Efron et al. (2004). Once the two constraints are established, we notice that

$$v_i = -\frac{\partial^2 C(y_i, F)}{\partial F^2}\bigg|_{F_0(\mathbf{x}_i)} u_i.$$

Hence we can plug the constraint 1 into the constraint 2 and get the following set of equations:

$$\tilde{H}_{\mathcal{A}}^T W \tilde{H}_{\mathcal{A}} \mathbf{P} = \lambda_{\mathcal{A}} \mathbf{1},$$

where

$$\begin{aligned} \tilde{H}_{\mathcal{A}} &= (\cdots sign(c_j)\mathbf{h}_j \cdots),\ j \in \mathcal{A}, \\ W &= diag\left(-\frac{\partial^2 C(y_i, F)}{\partial F^2}\Big|_{F_0(\mathbf{x}_i)} \right), \\ \mathbf{P} &= (\cdots P_j \cdots)',\ j \in \mathcal{A}. \end{aligned}$$

If $\tilde{H}$ is of rank $|\mathcal{A}|$ (we will get back to this issue in details in Appendix B), then $\mathbf{P}$, or equivalently $\mathbf{u}$ and $\mathbf{v}$, are uniquely determined up to a scale number.

Now we consider the l_1-constrained optimization problem (18)–(19). Let $\hat{\mathbf{F}}(s)$ be the fitted vector and $\hat{\mathbf{r}}(s)$ be the corresponding residual vector. Since $\hat{\mathbf{F}}(s)$ and $\hat{\mathbf{r}}(s)$ are smooth, define

$$\mathbf{u}^* \equiv \lim_{\Delta s \to 0} \frac{\hat{\mathbf{F}}(s+\Delta s) - \hat{\mathbf{F}}(s)}{\Delta s}, \tag{22}$$

$$\mathbf{v}^* \equiv \lim_{\Delta s \to 0} \frac{\hat{\mathbf{r}}(s+\Delta s) - \hat{\mathbf{r}}(s)}{\Delta s}. \tag{23}$$

Lemma 2 *Under the monotone coefficient paths assumption,* $\mathbf{u}^*$ *and* $\mathbf{v}^*$ *also satisfy constraints 1–2.*

Proof Write the coefficient β_j as $\beta_j^+ - \beta_j^-$, where

$$\begin{cases} \beta_j^+ = \beta_j, \beta_j^- = 0 & \text{if} \quad \beta_j > 0, \\ \beta_j^+ = 0, \beta_j^- = -\beta_j & \text{if} \quad \beta_j < 0. \end{cases}$$

The l_1-constrained optimization problem (18)–(19) is then equivalent to

$$\min_{\beta^+, \beta^-} \quad \sum_{i=1}^n C\left(y_i, h(\mathbf{x}_i)'(\beta^+ - \beta^-)\right), \tag{24}$$

$$\text{subject to} \quad \|\beta^+\|_1 + \|\beta^-\|_1 \le s, \beta^+ \ge 0, \beta^- \ge 0. \tag{25}$$

The corresponding Lagrangian dual is

$$L = \sum_{i=1}^n C\left(y_i, h(\mathbf{x}_i)'(\beta^+ - \beta^-)\right) + \lambda \sum_{j=1}^J (\beta_j^+ + \beta_j^-) \tag{26}$$

$$-\lambda \cdot s - \sum_{j=1}^J \lambda_j^+ \beta_j^+ - \sum_{j=1}^J \lambda_j^- \beta_j^-, \tag{27}$$

where $\lambda \ge 0, \lambda_j^+ \ge 0, \lambda_j^- \ge 0$ are Lagrange multipliers.

By differentiating the Lagrangian dual, we get the solution of (24)–(25) needed to satisfy the following Karush-Kuhn-Tucker conditions:

$$\frac{\partial L}{\partial \beta_j^+} = -\mathbf{h}_j' \hat{\mathbf{r}} + \lambda - \lambda_j^+ = 0, \tag{28}$$

$$\frac{\partial L}{\partial \beta_j^-} = \mathbf{h}_j'\hat{\mathbf{r}} + \lambda - \lambda_j^- = 0, \tag{29}$$

$$\lambda_j^+ \hat{\beta}_j^+ = 0, \tag{30}$$

$$\lambda_j^- \hat{\beta}_j^- = 0. \tag{31}$$

Let $c_j = \mathbf{h}_j'\hat{\mathbf{r}}$ and $\mathcal{A} = \{j : |c_j| = \max_j |c_j|\}$. We can see the following facts from the Karush-Kuhn-Tucker conditions:

(**Fact 1**) Use (28), (29) and $\lambda \geq 0, \lambda_j^+, \lambda_j^- \geq 0$, we have $|c_j| \leq \lambda$.

(**Fact 2**) If $\hat{\beta}_j \neq 0$, then $|c_j| = \lambda$ and $j \in \mathcal{A}$. For example, suppose $\hat{\beta}_j^+ \neq 0$, then $\lambda_j^+ = 0$ and (28) implies $c_j = \lambda$.

(**Fact 3**) If $\hat{\beta}_j \neq 0$, $sign(\hat{\beta}_j) = sign(c_j)$.

We also note that:

- $\hat{\beta}_j^+$ and $\hat{\beta}_j^-$ can not both be non-zero, otherwise $\lambda_j^+ = \lambda_j^- = 0$, (28) and (29) can not hold at the same time.
- It is possible that $\hat{\beta}_j = 0$ and $j \in \mathcal{A}$. This only happens for a finite number of s values, where basis $\mathbf{h}_j$ is about to enter the model.

For sufficiently small Δs, since the second derivative of the cost function $C(y,F)$ is finite, $\mathcal{A}$ will stay the same. Since $j \in \mathcal{A}$ if $\hat{\beta}_j \neq 0$, the change in the fitted vector is

$$\hat{\mathbf{F}}(s+\Delta s) - \hat{\mathbf{F}}(s) = \sum_{j \in \mathcal{A}} Q_j \mathbf{h}_j.$$

Since $sign(\hat{\beta}_j) = sign(c_j)$ and the coefficients $\hat{\beta}_j$ change monotonically, $sign(Q_j)$ will agree with $sign(c_j)$. Hence we have

$$\frac{\hat{\mathbf{F}}(s+\Delta s) - \hat{\mathbf{F}}(s)}{\Delta s} = \sum_{j \in \mathcal{A}} P_j sign(c_j)\mathbf{h}_j. \tag{32}$$

This implies $\mathbf{u}^*$ satisfies constraint 1. The claim $\mathbf{v}^*$ satisfies constraint 2 follows directly from fact 2, since both $\hat{\mathbf{r}}(s+\Delta s)$ and $\hat{\mathbf{r}}(s)$ satisfy constraint 2. ■

Completion of proof of Theorem (1): We further notice that in both the ε-boosting case and the constrained optimization case, we have $\sum_{j \in \mathcal{A}} P_j = 1$ by definition and the monotone coefficient paths condition, hence $\mathbf{u}$ and $\mathbf{v}$ are uniquely determined, i.e.:

$$\mathbf{u} = \mathbf{u}^* \text{ and } \mathbf{v} = \mathbf{v}^*.$$

To translate the result into $\hat{\beta}(s)$ and $\beta(T)$, we notice $F(\mathbf{x}) = h(\mathbf{x})'\beta$. Efron et al. (2004) showed that for $\nabla\hat{\beta}(s)$ to be well defined, $\mathcal{A}$ can have at most n elements, i.e., $|\mathcal{A}| \leq n$. We give sufficient conditions for when this is true in Appendix B.

Now Let

$$H_{\mathcal{A}} = (\cdots h_j(\mathbf{x}_i) \cdots), i = 1, \ldots n; j \in \mathcal{A}$$

be a $n \times |\mathcal{A}|$ matrix, which we assume is of rank $|\mathcal{A}|$. Then $\nabla\hat{\beta}(s)$ is given by

$$\nabla\hat{\beta}(s) = \left(H'_{\mathcal{A}} W H_{\mathcal{A}}\right)^{-1} H'_{\mathcal{A}} W \mathbf{u}^*,$$

and

$$\frac{\beta(T) - \hat{\beta}(s)}{T \cdot \varepsilon} \to \left(H'_{\mathcal{A}} W H_{\mathcal{A}}\right)^{-1} H'_{\mathcal{A}} W \mathbf{u}.$$

Hence the theorem is proved. ■

Appendix B. Uniqueness and Existence Results

In this appendix, we give some details on the properties of regularized solution paths. In section B.1 we formulate and prove sparseness and uniqueness results on l_1-regularized solutions for any convex loss. In section B.2 we extend Theorem 3 of Section 5—which proved the margin maximizing property of the limit of l_p-regularized solutions, as regularization varies—to the case that the margin maximizing solution is not unique.

B.1 Sparseness and Uniqueness of l_1-Regularized Solutions and Their Limits

Consider the l_1-constrained optimization problem:

$$\min_{\|\beta\|_1 \le c} \sum_{i=1}^{n} C(y_i, \beta' h(\mathbf{x}_i)). \tag{33}$$

In this section we give sufficient conditions for the following properties of the solutions of (33):

1. Existence of a sparse solution (with at most n non-zero coefficients),
2. Non-existence of non-sparse solutions with more than n non-zero coefficients,
3. Uniqueness of the solution,
4. Convergence of the solutions to sparse solution, as c increases.

Theorem 3 *Assume that the unconstrained solution for problem (33) has l_1 norm bigger than c. Then there exists a solution of (33) which has at most n non-zero coefficients.*

Proof As Lemma 2 in the Appendix A, we will prove the theorem using the Karush-Kuhn-Tucker (KKT) formulation of the optimization problem.

The chain rule for differentiation gives us that

$$\frac{\partial \sum_i C(y_i, \beta' h(\mathbf{x}_i))}{\partial \beta_j} = -\mathbf{h}'_j \mathbf{r}(\beta), \tag{34}$$

where $\mathbf{h}_j$ and $\mathbf{r}(\beta)$ are defined in the Appendix A; $\mathbf{r}(\beta)$ is the "generalized residual" vector. Using this simple relationship and fact 2 of Lemma 2 we can write a system of equations for all non-zero coefficients at the optimal constrained solution as follows (denote by $\mathcal{A}$ the set of indices for non-zero coefficients):

$$H'_{\mathcal{A}} \mathbf{r}(\beta) = \lambda \cdot \text{sign}\beta_{\mathcal{A}}. \tag{35}$$

In other words, we get $|\mathcal{A}|$ equations in $|\mathcal{A}|$ variables, corresponding to the non-zero β_j's.

However, each column of the matrix $H_\mathcal{A}$ is of length n, and so $H_\mathcal{A}$ can have at most n linearly independent columns, $\text{rank}(H_\mathcal{A}) \leq n$. Assume now that we have an optimal solution for (33) with $|\mathcal{A}| > n$. Then there exists $l \in \mathcal{A}$ such that

$$\mathbf{h}_l = \sum_{j \in \mathcal{A}, j \neq l} \alpha_j \mathbf{h}_j. \tag{36}$$

Substituting (36) into the l'th row in (35) we get

$$(\sum_{j \in \mathcal{A}, j \neq l} \alpha_j \mathbf{h}_j)' \mathbf{r}(\beta) = \lambda \cdot \text{sign}\beta_l. \tag{37}$$

But from (35) we know that $\mathbf{h}_j' \mathbf{r}(\beta) = \lambda \cdot \text{sign}\beta_j \ , \ \forall j \in \mathcal{A}$, meaning we can re-phrase (37) as

$$\sum_{j \in \mathcal{A}, j \neq l} \alpha_j \cdot \text{sign}\beta_j \cdot \text{sign}\beta_l = 1. \tag{38}$$

In other words, we get that $\mathbf{h}_l$ is a linear combination of the columns of $H_{\mathcal{A}-\{l\}}$ which must obey the specific numeric relation in (38).

Now we can construct an alternative optimal solution for (33) with one less non-zero coefficient, as follows:

1. Start from β
2. Define the direction γ in coefficient space implied by (36), that is:
   $\gamma_l = -\text{sign}\beta_l \ , \ \gamma_j = \alpha_j \cdot \text{sign}\beta_l \ , \ \forall j \in \mathcal{A} - \{l\}$
3. Move in direction γ until some coefficient in $\mathcal{A}$ hits zero, i.e., define:

   $$\delta^* = \min\left\{\delta > 0 \ : \ \exists j \in \mathcal{A} \text{ s.t. } \beta_j + \gamma_j \delta = 0\right\}$$

   (we know that $\delta^* \leq |\beta_l|$)
4. Set $\tilde{\beta} = \beta + \delta^* \gamma$

Then from (36) we get that $\tilde{\beta}' h(\mathbf{x}_i) = \beta' h(\mathbf{x}_i) \ , \ \forall i$ and from (38) we get that

$$\begin{aligned} \|\tilde{\beta}\|_1 &= \|\beta\|_1 - \sum_{j \in \mathcal{A}} [|\beta_j + \gamma_j \delta^*| - |\beta_j|] = \\ &= \|\beta\|_1 - \delta^* \cdot \left(1 - \sum_{j \in \mathcal{A}-l} \alpha_j \cdot \text{sign}\beta_j \text{sign}\beta_l\right) = \|\beta\|_1. \end{aligned} \tag{39}$$

So $\tilde{\beta}$ generates the same fit as β and has the same l_1 norm, therefore it is also an optimal solution, with at least one less non-zero coefficient (from the definition of δ^*).

We can obviously apply this process repeatedly until we get a solution with at most n non-zero coefficients. ■

This theorem has the following immediate implication:

Corollary 4 *If there is no set of more than n dictionary functions which obeys the equalities (36,38) on the training data, then* any *solution of (33) has at most n non-zero coefficients.*

This corollary implies, for example, that if the basis functions come from a "continuous non-redundant" distribution (which means that any equality would hold with probability 0) then with probability 1 any solution of (33) has at most n non-zero coefficients.

Theorem 5 *Assume that there is no set of more than n dictionary functions which obeys the equalities (36,38) on the training data. In addition assume:*

1. *The loss function C is strictly convex (squared error loss, C_l and C_e obviously qualify),*
2. *No set of dictionary functions of size $\leq n$ is linearly dependent on the training data.*

Then the problem (33) has a unique solution.

Proof The previous corollary tells us that any solution has at most n non-zero coefficients. Now assume $\beta_1,\ \beta_2$ are both solutions of (33). From strict convexity of the loss we get that

$$h(X)'\beta_1 = h(X)'\beta_2 = h(X)'(\alpha\beta_1 + (1-\alpha)\beta_2)\ ,\ \ \forall 0 \leq \alpha \leq 1; \tag{40}$$

and from convexity of the l_1 norm we get

$$\|\alpha\beta_1 + (1-\alpha)\beta_2\|_1 \leq \|\beta_1\|_1 = \|\beta_2\|_1 = c. \tag{41}$$

So $(\alpha\beta_1 + (1-\alpha)\beta_2)$ must also be a solution. Thus, the total number of variables with non-zero coefficients in either β_1 or β_2 cannot be bigger than n, since then $(\alpha\beta_1 + (1-\alpha)\beta_2)$, would have $> n$ non-zero coefficients for almost all values of α, contradicting Corollary 4. Thus, by ignoring all coefficients which are 0 in both β_1 and β_2 we get that both β_1 and β_2 can be represented in the same $n-dimensional$ (maximum) sub-space of $\mathbb{R}^J$. Which leads to a contradiction between (40) and assumption 2. ■

Corollary 6 *Consider a sequence $\{\frac{\hat{\beta}(c)}{c} : 0 \leq c \leq \infty\}$ of normalized solutions to the problem (33). Assume that all these solutions have at most n non-zero coefficients. Then any limit point of the sequence has at most n non-zero coefficients.*

Proof This is a trivial consequence of convergence. Assume by contradiction β^* is a convergence point with more than n non-zero coefficients. Let $k = \arg\min_j\{|\beta_j^*| : \beta_j^* \neq 0\}$. Then for *any* vector $\tilde{\beta}$ with at most n non-zero coefficients we know that $\|\tilde{\beta} - \beta^*\| \geq |\beta_j^*| > 0$ so we get a contradiction to convergence. ■

B.2 Uniqueness of Limiting Solution in Theorem 3 when Margin Maximizing Separator is not Unique

Recall, that we are interested in convergence points of the normalized regularized solutions $\frac{\hat{\beta}^{(p)}(c)}{c}$. Theorem 3 proves that any such convergence point corresponds to an l_p-margin maximizing separating hyper-plane. We now extend it to the case that this first-order separator is not unique, by extending the result to consider the second smallest margin as a "tie breaker". We show that any convergence point maximizes the second smallest margin among all models with maximal minimal margin. If there are also ties in the second smallest margin, then any limit point maximizes the third smallest margin among all models which still remain, and so on. It should be noted that the minimal margin is typically not attained by one observation only in margin maximizing models. In case of ties in the smallest margins our reference to "smallest", "second smallest" etc. implies arbitrary tie-breaking (i.e., our decision on which one of the tied margins is considered smallest, and which one second smallest is of no consequence).

Theorem 7 *Assume that the data is separable and that the margin-maximizing separating hyper-plane, as defined in (4) is not unique. Then any convergence point of* $\frac{\hat{\beta}^{(p)}(c)}{c}$ *will correspond to a margin-maximizing separating hyper-plane which also maximizes the second smallest margin.*

Proof The proof is essentially the same as that of Theorem 3. We outline it below.

From Theorem 3 we know that we only need to consider margin-maximizing models as limit points. Thus let β_1, β_2 be two margin maximizing models with l_p norm 1, but let β_1 have a bigger *second smallest* margin. Assume that β_1 attains its smallest margin on observation i_1 and β_2 attains the same smallest margin on observation i_2. Now define

$$m_1 = \min_{i \neq i_1} y_i h(\mathbf{x}_i)' \beta_1 > \min_{i \neq i_2} y_i h(\mathbf{x}_i)' \beta_2 = m_2.$$

Then we have that Lemma 4 of Theorem 3 holds for β_1 and β_2 (the proof is exactly the same, except that we ignore the smallest margin observation for each model, since these always contribute the same amount to the combined loss).

Let β^* be a convergence point. We know β^* maximizes the margin from Theorem 3. Now assume $\tilde{\beta}$ also maximizes the margin but has bigger second-smallest margin than β^*. Then we can proceed exactly as the proof of Theorem 3, considering only $n-1$ observations for each model and using our modified Lemma 4, to conclude that β^* cannot be a convergence point (again note that the smallest margin observation always contributes the same to the loss of both models). ■

In the case that the two smallest margins still do not define a unique solution, we can continue up the list of margins, applying this result recursively. The conclusion is that the limit of the normalized, l_p -regularized models "maximizes the margins", and not just the minimal margin. The only case when this convergence point is not unique is, therefore, the case that the whole order statistic of the optimal separator is not unique. It is an interesting research question to investigate under which conditions this scenario is possible.

References

C. L. Blake and C. J. Merz. Repository of machine learning databases. [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science., 1998.

L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Computational Learing Theory*, pages 158–169, 2000.

D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *J. R. Statist. Soc. B.*, 57(2):301–337, 1995.

B. Efron, T. Hastie, I. M. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2), 2004.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 2001.

J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.

T. Hastie, T. Tibshirani, and J. H. Friedman. *Elements of Statistical Learning*. Springer-Verlag, New York, 2001.

O. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24(1–2):15–23, 1999.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, volume 12, 1999.

G. Rätsch, S. Mika, and M. K. Warmuth. On the convergence of leveraging. NeuroCOLT2 Technical Report 98, Royal Holloway College, London, 2001a.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3): 287–320, March 2001b.

G. Rätsch and M. W. Warmuth. Efficient margin maximization with boosting. submitted to JMLR, December 2002.

S. Rosset and E. Segal. Boosting density estimation. NIPS-02, 2003.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26:1651–1686, 1998.

T. Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, 49, 2003.

T. Zhang and B. Yu. Boosting with early stopping: Convergence and results. Techincal report, Dept. of Statistics, Univ. of California, Berkeley, 2003.

Journal of Machine Learning Research 5 (2004) 975–1005 Submitted 11/03; Revised 05/04; Published 8/04

Probability Estimates for Multi-class Classification by Pairwise Coupling

Ting-Fan Wu B89098@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Ruby C. Weng CHWENG@NCCU.EDU.TW
Department of Statistics
National Chengchi University
Taipei 116, Taiwan

Editor: Yoram Singer

Abstract

Pairwise coupling is a popular multi-class classification method that combines all comparisons for each pair of classes. This paper presents two approaches for obtaining class probabilities. Both methods can be reduced to linear systems and are easy to implement. We show conceptually and experimentally that the proposed approaches are more stable than the two existing popular methods: voting and the method by Hastie and Tibshirani (1998).

Keywords: pairwise coupling, probability estimates, random forest, support vector machines

1. Introduction

The multi-class classification problem refers to assigning each of the observations into one of k classes. As two-class problems are much easier to solve, many authors propose to use two-class classifiers for multi-class classification. In this paper we focus on techniques that provide a multi-class probability estimate by combining all pairwise comparisons.

A common way to combine pairwise comparisons is by voting (Knerr et al., 1990; Friedman, 1996). It constructs a rule for discriminating between every pair of classes and then selecting the class with the most winning two-class decisions. Though the voting procedure requires just pairwise decisions, it only predicts a class label. In many scenarios, however, probability estimates are desired. As numerous (pairwise) classifiers do provide class probabilities, several authors (Refregier and Vallet, 1991; Price et al., 1995; Hastie and Tibshirani, 1998) have proposed probability estimates by combining the pairwise class probabilities.

Given the observation $\mathbf{x}$ and the class label y, we assume that the estimated pairwise class probabilities r_{ij} of $\mu_{ij} = P(y = i \mid y = i \text{ or } j, \mathbf{x})$ are available. From the ith and jth classes of a training set, we obtain a model which, for any new $\mathbf{x}$, calculates r_{ij} as an approximation of μ_{ij}. Then, using all r_{ij}, the goal is to estimate $p_i = P(y = i \mid \mathbf{x}), i = 1, \ldots, k$. In this paper, we first propose a method for obtaining probability estimates via an approxima-

tion solution to an identity. The existence of the solution is guaranteed by theory in finite Markov Chains. Motivated by the optimization formulation of this method, we propose a second approach. Interestingly, it can also be regarded as an improved version of the coupling approach given by Refregier and Vallet (1991). Both of the proposed methods can be reduced to solving linear systems and are simple in practical implementation. Furthermore, from conceptual and experimental points of view, we show that the two proposed methods are more stable than voting and the method by Hastie and Tibshirani (1998).

We organize the paper as follows. In Section 2, we review several existing methods. Sections 3 and 4 detail the two proposed approaches. Section 5 presents the relationship between different methods through their corresponding optimization formulas. In Section 6, we compare these methods using simulated data. In Section 7, we conduct experiments using real data. The classifiers considered are support vector machines and random forest. A preliminary version of this paper was presented previously (Wu et al., 2004).

2. Survey of Existing Methods

For methods surveyed in this section and those proposed later, each provides a vector of multi-class probability estimates. We denote it as $\mathbf{p}^*$ according to method $*$. Similarly, there is an associated rule $\arg\max_i[p_i^*]$ for prediction and we denote the rule as δ_*.

2.1 Voting

Let r_{ij} be the estimates of $\mu_{ij} \equiv P(y = i \mid y = i \text{ or } j, \mathbf{x})$ and assume $r_{ij} + r_{ji} = 1$. The voting rule (Knerr et al., 1990; Friedman, 1996) is

$$\delta_V = \arg\max_i [\sum_{j:j\neq i} I_{\{r_{ij}>r_{ji}\}}], \tag{1}$$

where I is the indicator function: $I\{x\} = 1$ if x is true, and 0 otherwise. A simple estimate of probabilities can be derived as

$$p_i^v = 2 \sum_{j:j\neq i} I_{\{r_{ij}>r_{ji}\}}/(k(k-1)).$$

2.2 Method by Refregier and Vallet

With $\mu_{ij} = p_i/(p_i + p_j)$, Refregier and Vallet (1991) consider that

$$\frac{r_{ij}}{r_{ji}} \approx \frac{\mu_{ij}}{\mu_{ji}} = \frac{p_i}{p_j}. \tag{2}$$

Thus, making (2) an equality may be a way to solve p_i. However, the number of equations, $k(k-1)/2$, is more than the number of unknowns k, so Refregier and Vallet (1991) propose to choose any $k-1$ r_{ij}. Then, with the condition $\sum_{i=1}^k p_i = 1$, $\mathbf{p}^{RV}$ can be obtained by solving a linear system. However, as pointed out previously by Price et al. (1995), the results depend strongly on the selection of $k-1$ r_{ij}.

In Section 4, by considering (2) as well, we propose a method which remedies this problem.

2.3 Method by Price, Knerr, Personnaz, and Dreyfus

Price et al. (1995) consider that

$$\left(\sum_{j:j\neq i} P(y=i \text{ or } j \mid \mathbf{x})\right) - (k-2)P(y=i \mid \mathbf{x}) = \sum_{j=1}^{k} P(y=j \mid \mathbf{x}) = 1.$$

Using

$$r_{ij} \approx \mu_{ij} = \frac{P(y=i \mid \mathbf{x})}{P(y=i \text{ or } j \mid \mathbf{x})},$$

one obtains

$$p_i^{PKPD} = \frac{1}{\sum_{j:j\neq i} \frac{1}{r_{ij}} - (k-2)}. \tag{3}$$

As $\sum_{i=1}^{k} p_i = 1$ does not hold, we must normalize $\mathbf{p}^{PKPD}$. This approach is very simple and easy to implement. In the rest of this paper, we refer to this method as PKPD.

2.4 Method by Hastie and Tibshirani

Hastie and Tibshirani (1998) propose to minimize the Kullback-Leibler (KL) distance between r_{ij} and μ_{ij}:

$$\begin{aligned} l(\mathbf{p}) &= \sum_{i\neq j} n_{ij} r_{ij} \log \frac{r_{ij}}{\mu_{ij}}, \qquad (4) \\ &= \sum_{i<j} n_{ij} \left(r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + (1-r_{ij}) \log \frac{1-r_{ij}}{1-\mu_{ij}} \right), \end{aligned}$$

where $\mu_{ij} = p_i/(p_i+p_j)$, $r_{ji} = 1 - r_{ij}$, and n_{ij} is the number of training data in the ith and jth classes.

To minimize (4), they first calculate

$$\frac{\partial l(\mathbf{p})}{\partial p_i} = \sum_{j:j\neq i} n_{ij} \left(-\frac{r_{ij}}{p_i} + \frac{1}{p_i+p_j}\right).$$

Thus, letting $\partial l(\mathbf{p})/\partial p_i = 0, i = 1, \ldots, k$ and multiplying p_i on each term, Hastie and Tibshirani (1998) propose finding a point that satisfies

$$\sum_{j:j\neq i} n_{ij}\mu_{ij} = \sum_{j:j\neq i} n_{ij} r_{ij}, \qquad \sum_{i=1}^{k} p_i = 1, \text{and} \qquad p_i > 0, i = 1, \ldots, k. \tag{6}$$

Such a point is obtained by the following algorithm:

Algorithm 1

1. *Start with some initial $p_j > 0, \forall j$ and corresponding $\mu_{ij} = p_i/(p_i+p_j)$.*

2. *Repeat* $(i = 1, \ldots, k, 1, \ldots)$

$$\alpha = \frac{\sum_{j:j\neq i} n_{ij} r_{ij}}{\sum_{j:j\neq i} n_{ij}\mu_{ij}} \tag{7}$$

$$\mu_{ij} \leftarrow \frac{\alpha\mu_{ij}}{\alpha\mu_{ij} + \mu_{ji}}, \ \ \mu_{ji} \leftarrow 1 - \mu_{ij}, \ \textit{for all } j \neq i \tag{8}$$

$$p_i \leftarrow \alpha p_i \tag{9}$$

$$\textit{normalize } \mathbf{p} \ (\textit{optional}) \tag{10}$$

until k *consecutive* α *are all close to ones.*

3. $\mathbf{p} \leftarrow \mathbf{p}/\sum_{i=1}^{k} p_i$

(9) implies that in each iteration, only the ith component is updated and all others remain the same. There are several remarks about this algorithm. First, the initial $\mathbf{p}$ must be positive so that all later $\mathbf{p}$ are positive and α is well defined (i.e., no zero denominator in (7)). Second, (10) is an optional operation because whether we normalize $\mathbf{p}$ or not does not affect the values of μ_{ij} and α in (7) and (8).

Hastie and Tibshirani (1998) prove that Algorithm 1 generates a sequence of points at which the KL distance is strictly decreasing. However, Hunter (2004) indicates that the strict decrease in $l(\mathbf{p})$ does not guarantee that any limit point satisfies (6). Hunter (2004) discusses the convergence of algorithms for generalized Bradley-Terry models where Algorithm 1 is a special case. It points out that Zermelo (1929) has proved that, if $r_{ij} > 0, \forall i \neq j$, for any initial point, the whole sequence generated by Algorithm 1 converges to a point satisfying (6). Furthermore, this point is the unique global minimum of $l(\mathbf{p})$ under the constraints $\sum_{i=1}^{k} p_i = 1$ and $0 \leq p_i \leq 1, i = 1, \ldots, k$.

Let $\mathbf{p}^{HT}$ denote the global minimum of $l(\mathbf{p})$. It is shown in Zermelo (1929) and Theorem 1 of (Hastie and Tibshirani, 1998) that if weights n_{ij} in (4) are considered equal, then $\mathbf{p}^{HT}$ satisfies

$$p_i^{HT} > p_j^{HT} \text{ if and only if } \tilde{p}_i^{HT} \equiv \frac{2\sum_{s:i\neq s} r_{is}}{k(k-1)} > \tilde{p}_j^{HT} \equiv \frac{2\sum_{s:j\neq s} r_{js}}{k(k-1)}. \tag{11}$$

Therefore, $\tilde{\mathbf{p}}^{HT}$ is sufficient if one only requires the classification rule. In fact, $\tilde{\mathbf{p}}^{HT}$ can be derived as an approximation to the identity

$$p_i = \sum_{j:j\neq i} \left(\frac{p_i + p_j}{k-1}\right)\left(\frac{p_i}{p_i + p_j}\right) = \sum_{j:j\neq i} \left(\frac{p_i + p_j}{k-1}\right)\mu_{ij} \tag{12}$$

by replacing $p_i + p_j$ with $2/k$, and μ_{ij} with r_{ij} in (12). We refer to the decision rule as δ_{HT}, which is essentially

$$\arg\max_i [\tilde{p}_i^{HT}]. \tag{13}$$

In the next two sections, we propose two methods which are simpler in both practical implementation and algorithmic analysis.

If the multi-class data are balanced, it is reasonable to assume equal weighting (i.e., $n_{ij} = 1$) as the above. In the rest of this paper, we restrict our discussion under such an assumption.

3. Our First Approach

As δ_{HT} relies on $p_i + p_j \approx 2/k$, in Section 6 we use two examples to illustrate possible problems with this rule. In this section, instead of replacing $p_i + p_j$ by $2/k$ in (12), we propose to solve the system:

$$p_i = \sum_{j:j\neq i} (\frac{p_i + p_j}{k-1}) r_{ij}, \forall i, \qquad \text{subject to } \sum_{i=1}^{k} p_i = 1, p_i \geq 0, \forall i. \tag{14}$$

Let $\mathbf{p}^1$ denote the solution to (14). Then the resulting decision rule is

$$\delta_1 = \arg\max_i [p_i^1].$$

3.1 Solving (14)

To solve (14), we rewrite it as

$$Q\mathbf{p} = \mathbf{p}, \quad \sum_{i=1}^{k} p_i = 1, \quad p_i \geq 0, \forall i, \text{ where } Q_{ij} \equiv \begin{cases} r_{ij}/(k-1) & \text{if } i \neq j, \\ \sum_{s:s\neq i} r_{is}/(k-1) & \text{if } i = j. \end{cases} \tag{15}$$

Observe that $\sum_{i=1}^{k} Q_{ij} = 1$ for $j = 1, \ldots, k$ and $0 \leq Q_{ij} \leq 1$ for $i, j = 1, \ldots, k$, so there exists a finite Markov Chain whose transition matrix is Q. Moreover, if $r_{ij} > 0$ for all $i \neq j$, then $Q_{ij} > 0$, which implies that this Markov Chain is irreducible and aperiodic. From Theorem 4.3.3 of Ross (1996), these conditions guarantee the existence of a unique stationary probability and all states being positive recurrent. Hence, we have the following theorem:

Theorem 1 *If $r_{ij} > 0$, $i \neq j$, then (15) has a unique solution $\mathbf{p}$ with $0 < p_i < 1$ $\forall i$.*

Assume the solution from Theorem 1 is $\mathbf{p}^*$. We claim that without the constraints $p_i \geq 0, \forall i$, the linear system

$$Q\mathbf{p} = \mathbf{p}, \qquad \sum_{i=1}^{k} p_i = 1 \tag{16}$$

still has the same unique solution $\mathbf{p}^*$. Otherwise, there is another solution $\bar{\mathbf{p}}^* (\neq \mathbf{p}^*)$. Then for any $0 \leq \lambda \leq 1$, $\lambda\mathbf{p}^* + (1-\lambda)\bar{\mathbf{p}}^*$ satisfies (16) as well. As $p_i^* > 0, \forall i$, when λ is sufficiently close to 1, $\lambda p_i^* + (1-\lambda)\bar{p}_i^* > 0, i = 1, \ldots, k$. This violates the uniqueness property in Theorem 1.

Therefore, unlike the method in Section 2.4 where a special iterative procedure has to be implemented, here we only solve a simple linear system. As (16) has $k+1$ equalities but only k variables, practically we remove any one equality from $Q\mathbf{p} = \mathbf{p}$ and obtain a square system. Since the column sum of Q is the vector of all ones, the removed equality is a linear combination of all remaining equalities. Thus, any solution of the square system satisfies (16) and vice versa. Therefore, this square system has the same unique solution as (16) and hence can be solved by standard Gaussian elimination.

Instead of Gaussian elimination, as the stationary solution of a Markov Chain can be derived by the limit of the n-step transition probability matrix Q^n, we can solve (14) by repeatedly multiplying Q with any initial probability vector.

3.2 Another Look at (14)

The following arguments show that the solution to (14) is a global minimum of a meaningful optimization problem. To begin, using the property that $r_{ij}+r_{ji}=1$, $\forall i \neq j$, we re-express $p_i = \sum_{j:j\neq i}(\frac{p_i+p_j}{k-1})r_{ij}$ of (15) (i.e., $Q\mathbf{p}=\mathbf{p}$ of (16)) as

$$\sum_{j:j\neq i} r_{ji}p_i - \sum_{j:j\neq i} r_{ij}p_j = 0, i = 1,\ldots,k.$$

Therefore, a solution of (15) is in fact the unique global minimum of the following convex problem:

$$\begin{aligned} \min_{\mathbf{p}} \quad & \sum_{i=1}^{k}(\sum_{j:j\neq i} r_{ji}p_i - \sum_{j:j\neq i} r_{ij}p_j)^2 \\ \text{subject to} \quad & \sum_{i=1}^{k} p_i = 1, \qquad p_i \geq 0, i = 1,\ldots,k. \end{aligned} \tag{18}$$

The reason is that the object function is always nonnegative, and it attains zero under (15). Note that the constraints $p_i \geq 0$ $\forall i$ are not redundant following the discussion around Equation (16).

4. Our Second Approach

Note that both approaches in Sections 2.4 and 3 involve solving optimization problems using relations like $p_i/(p_i+p_j) \approx r_{ij}$ or $\sum_{j:j\neq i} r_{ji}p_i \approx \sum_{j:j\neq i} r_{ij}p_j$. Motivated by (18), we suggest another optimization formulation as follows:

$$\min_{\mathbf{p}} \sum_{i=1}^{k}\sum_{j:j\neq i}(r_{ji}p_i - r_{ij}p_j)^2 \qquad \text{subject to } \sum_{i=1}^{k} p_i = 1, p_i \geq 0, \forall i. \tag{19}$$

Note that the method (Refregier and Vallet, 1991) described in Section 2.2 considers a random selection of $k-1$ equations of the form $r_{ji}p_i = r_{ij}p_j$. As (19) considers all $r_{ij}p_j - r_{ji}p_i$, not just $k-1$ of them, it can be viewed as an improved version of the coupling approach by Refregier and Vallet (1991).

Let $\mathbf{p}^2$ denote the corresponding solution. We then define the classification rule as

$$\delta_2 = \arg\max_i [p_i^2].$$

4.1 A Linear System from (19)

Since (18) has a unique solution, which can be obtained by solving a simple linear system, it is desirable to see whether the minimization problem (19) has these nice properties. In this subsection, we show that (19) has a unique solution and can be solved by a simple linear system.

First, the following theorem shows that the nonnegative constraints in (19) are redundant.

Theorem 2 *Problem* (19) *is equivalent to*

$$\min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j\neq i} (r_{ji}p_i - r_{ij}p_j)^2 \qquad \text{subject to} \sum_{i=1}^{k} p_i = 1. \tag{20}$$

The proof is in Appendix A. Note that we can rewrite the objective function of (20) as

$$\min_{\mathbf{p}} \ \ 2\mathbf{p}^T Q \mathbf{p} \equiv \min_{\mathbf{p}} \ \ \frac{1}{2}\mathbf{p}^T Q \mathbf{p}, \tag{21}$$

where

$$Q_{ij} = \begin{cases} \sum_{s:s\neq i} r_{si}^2 & \text{if } i = j, \\ -r_{ji}r_{ij} & \text{if } i \neq j. \end{cases} \tag{22}$$

We divide the objective function by a positive factor of four so its derivative is a simple form $Q\mathbf{p}$. From (22), Q is positive semi-definite as for any $\mathbf{v} \neq \mathbf{0}, \mathbf{v}^T Q \mathbf{v} = 1/2 \sum_{i=1}^{k} \sum_{j=1}^{k} (r_{ji}v_i - r_{ij}v_j)^2 \geq 0$. Therefore, without constraints $p_i \geq 0, \forall i$, (21) is a linear-equality-constrained convex quadratic programming problem. Consequently, a point $\mathbf{p}$ is a global minimum if and only if it satisfies the optimality condition: There is a scalar b such that

$$\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}. \tag{23}$$

Here $Q\mathbf{p}$ is the derivative of (21), b is the Lagrangian multiplier of the equality constraint $\sum_{i=1}^{k} p_i = 1$, $\mathbf{e}$ is the $k \times 1$ vector of all ones, and $\mathbf{0}$ is the $k \times 1$ vector of all zeros. Thus, the solution to (19) can be obtained by solving the simple linear system (23).

4.2 Solving (23)

Equation (23) can be solved by some direct methods in numerical linear algebra. Theorem 3(i) below shows that the matrix in (23) is invertible; therefore, Gaussian elimination can be easily applied.

For symmetric positive definite systems, Cholesky factorization reduces the time for Gaussian elimination by half. Though (23) is symmetric but not positive definite, if Q is positive definite, Cholesky factorization can be used to obtain $b = -1/(\mathbf{e}^T Q^{-1} \mathbf{e})$ first and then $\mathbf{p} = -bQ^{-1}\mathbf{e}$. Theorem 3(ii) shows that Q is positive definite under quite general conditions. Moreover, even if Q is only positive semi-definite, Theorem 3(i) proves that $Q + \Delta \mathbf{e}\mathbf{e}^T$ is positive definite for any constant $\Delta > 0$. Along with the fact that (23) is equivalent to

$$\begin{bmatrix} Q + \Delta \mathbf{e}\mathbf{e}^T & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ b \end{bmatrix} = \begin{bmatrix} \Delta \mathbf{e} \\ 1 \end{bmatrix},$$

we can do Cholesky factorization on $Q + \Delta \mathbf{e}\mathbf{e}^T$ and solve b and $\mathbf{p}$ similarly, regardless whether Q is positive definite or not.

Theorem 3 *If* $r_{tu} > 0 \ \forall t \neq u$, *we have*

(i) For any $\Delta > 0$, $Q + \Delta \mathbf{e}\mathbf{e}^T$ is positive definite. In addition, $\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is invertible, and hence (19) has a unique global minimum.

(ii) If for any $i = 1, \ldots, k$, there are $s \neq j$ for which $s \neq i$, $j \neq i$, and

$$\frac{r_{si} r_{sj}}{r_{is}} \neq \frac{r_{ji} r_{js}}{r_{ij}}, \tag{25}$$

then Q is positive definite.

We leave the proof in Appendix B.

In addition to direct methods, next we propose a simple iterative method for solving (23):

Algorithm 2

1. *Start with some initial $p_i \geq 0, \forall i$ and $\sum_{i=1}^k p_i = 1$.*

2. *Repeat $(t = 1, \ldots, k, 1, \ldots)$*

$$p_t \leftarrow \frac{1}{Q_{tt}}[-\sum_{j:j \neq t} Q_{tj} p_j + \mathbf{p}^T Q \mathbf{p}] \tag{26}$$

$$\textit{normalize } \mathbf{p} \tag{27}$$

until (23) is satisfied.

Equation (22) and the assumption $r_{ij} > 0, \forall i \neq j$, ensure that the right-hand side of (26) is always nonnegative. For (27) to be well defined, we must ensure that $\sum_{i=1}^k p_i > 0$ after the operation in (26). This property holds (see (43) for more explanation). With $b = -\mathbf{p}^T Q \mathbf{p}$ obtained from (23), (26) is motivated from the tth equality in (23) with b replaced by $-\mathbf{p}^T Q \mathbf{p}$. The convergence of Algorithm 2 is established in the following theorem:

Theorem 4 *If $r_{sj} > 0, \forall s \neq j$, then $\{\mathbf{p}^i\}_{i=1}^{\infty}$, the sequence generated by Algorithm 2, converges globally to the unique minimum of (19).*

The proof is in Appendix C. Algorithm 2 is implemented in the software LIBSVM developed by Chang and Lin (2001) for multi-class probability estimates. We discuss some implementation issues of Algorithm 2 in Appendix D.

5. Relations Between Different Methods

Among the methods discussed in this paper, the four decision rules δ_{HT}, δ_1, δ_2, and δ_V can be written as $\arg\max_i[p_i]$, where $\mathbf{p}$ is derived by the following four optimization formulations

under the constraints $\sum_{i=1}^{k} p_i = 1$ and $p_i \geq 0, \forall i$:

$$\delta_{HT} : \min_{\mathbf{p}} \sum_{i=1}^{k} [\sum_{j:j\neq i} (r_{ij}\frac{1}{k} - \frac{1}{2}p_i)]^2, \tag{28}$$

$$\delta_1 : \quad \min_{\mathbf{p}} \sum_{i=1}^{k} [\sum_{j:j\neq i} (r_{ij}p_j - r_{ji}p_i)]^2, \tag{29}$$

$$\delta_2 : \quad \min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j\neq i} (r_{ij}p_j - r_{ji}p_i)^2, \tag{30}$$

$$\delta_V : \quad \min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j\neq i} (I_{\{r_{ij}>r_{ji}\}}p_j - I_{\{r_{ji}>r_{ij}\}}p_i)^2. \tag{31}$$

Note that (28) can be easily verified from (11), and that (29) and (30) have been explained in Sections 3 and 4. For (31), its solution is

$$p_i = \frac{c}{\sum_{j:j\neq i} I_{\{r_{ji}>r_{ij}\}}}, \tag{32}$$

where c is the normalizing constant; and therefore, $\arg\max_i[p_i]$ is the same as (1).[1] Detailed derivation of (32) is in Appendix E.

Clearly, (28) can be obtained from (29) by letting $p_j = 1/k$ and $r_{ji} = 1/2$. Such approximations ignore the differences between p_i. Next, (31) is from (30) with r_{ij} replaced by $I_{\{r_{ij}>r_{ji}\}}$, and hence, (31) may enlarge the differences between p_i. Moreover, compared with (30), (29) allows the difference between $r_{ij}p_j$ and $r_{ji}p_i$ to be canceled first, so (29) may tend to underestimate the differences between p_i. In conclusion, conceptually, (28) and (31) are more extreme – the former tends to underestimate the differences between p_i, while the latter overestimates them. These arguments will be supported by simulated and real data in the next two sections.

For PKPD approach (3), the decision rule can be written as:

$$\delta_{PKPD} = \arg\min_i [\sum_{j:j\neq i} \frac{1}{r_{ij}}].$$

This form looks similar to $\delta_{HT} = \arg\max_i[\sum_{j:j\neq i} r_{ij}]$, which can be obtained from (11) and (13). Notice that the differences among $\sum_{j:j\neq i} r_{ij}$ tend to be larger than those among $\sum_{j:j\neq i} \frac{1}{r_{ij}}$, because $1/r_{ij} > 1 > r_{ij}$. More discussion on these two rules will be given in Section 6.

6. Experiments on Synthetic Data

In this section, we use synthetic data to compare the performance of existing methods described in Section 2 as well as two new approaches proposed in Sections 3 and 4. Here we

1. For $I_{\{r_{ij}>r_{ji}\}}$ to be well defined, we consider $r_{ij} \neq r_{ji}$, which is generally true. In addition, if there is an i for which $\sum_{j:j\neq i} I_{\{r_{ji}>r_{ij}\}} = 0$, an optimal solution of (31) is $p_i = 1$, and $p_j = 0, \forall j \neq i$. The resulting decision is the same as that of (1).

do not include the method in Section 2.2 because its results depend strongly on the choice of $k-1$ r_{ij} and our second method is an improved version of it.

Hastie and Tibshirani (1998) design a simple experiment in which all p_i are fairly close and their method δ_{HT} outperforms the voting strategy δ_V. We conduct this experiment first to assess the performance of our proposed methods. Following their settings, we define class probabilities

(a) $p_1 = 1.5/k$, $p_j = (1-p_1)/(k-1)$, $j = 2, \ldots, k$,

and then set

$$r_{ij} = \frac{p_i}{p_i + p_j} + 0.1 z_{ij} \qquad \text{if } i > j, \tag{33}$$

$$r_{ji} = \frac{p_j}{p_i + p_j} + 0.1 z_{ji} = 1 - r_{ij} \text{ if } j > i, \tag{34}$$

where z_{ij} are standard normal variates and $z_{ji} = -z_{ij}$. Since r_{ij} are required to be within (0,1), we truncate r_{ij} at ϵ below and $1-\epsilon$ above, with $\epsilon = 10^{-7}$. In this example, class 1 has the highest probability and hence is the correct class.

Figure 2(a) shows accuracy rates for each of the five methods when $k = 2^2, \lceil 2^{2.5} \rceil, 2^3, \ldots, 2^7$, where $\lceil x \rceil$ denotes the largest integer not exceeding x. The accuracy rates are averaged over 1,000 replicates. Note that in this experiment all classes are quite competitive, so, when using δ_V, sometimes the highest vote occurs at two or more different classes. We handle this problem by randomly selecting one class from the ties. This partly explains the poor performance of δ_V. Another explanation is that the r_{ij} here are all close to 1/2, but (31) uses 1 or 0 instead, as stated in the previous section; therefore, the solution may be severely biased. Besides δ_V, the other four rules have good performance in this example.

Since δ_{HT} relies on the approximation $p_i + p_j \approx k/2$, this rule may suffer some losses if the class probabilities are not highly balanced. To examine this point, we consider the following two sets of class probabilities:

(b) We let $k_1 = k/2$ if k is even, and $(k+1)/2$ if k is odd; then we define $p_1 = 0.95 \times 1.5/k_1$, $p_i = (0.95 - p_1)/(k_1 - 1)$ for $i = 2, \ldots, k_1$, and $p_i = 0.05/(k - k_1)$ for $i = k_1 + 1, \ldots, k$.

(c) We define $p_1 = 0.95 \times 1.5/2$, $p_2 = 0.95 - p_1$, and $p_i = 0.05/(k-2), i = 3, \ldots, k$.

An illustration of these three sets of class probabilities is in Figure 1.

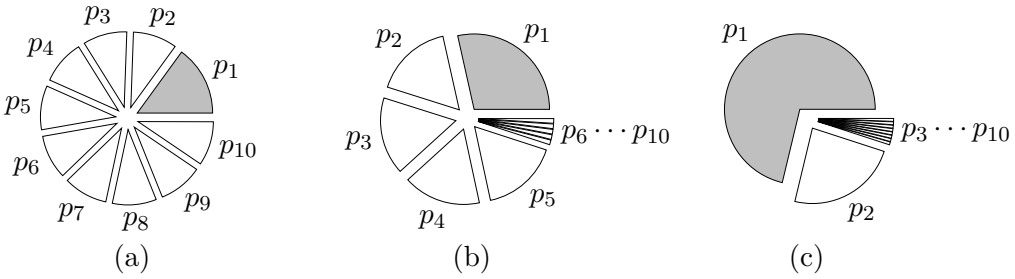


Figure 1: Three sets of class probabilities

After setting p_i, we define the pairwise comparisons r_{ij} as in (33)-(34). Both experiments are repeated for 1,000 times. The accuracy rates are shown in Figures 2(b) and 2(c). In both scenarios, p_i are not balanced. As expected, δ_{HT} is quite sensitive to the imbalance of p_i. The situation is much worse in Figure 2(c) because the approximation $p_i + p_j \approx k/2$ is more seriously violated, especially when k is large.

A further analysis of Figure 2(c) shows that when k is large,

$$\begin{aligned} r_{12} &= \frac{3}{4} + 0.1z_{12}, r_{1j} \approx 1 + 0.1z_{1j}, j \geq 3, \\ r_{21} &= \frac{1}{4} + 0.1z_{21}, r_{2j} \approx 1 + 0.1z_{2j}, j \geq 3, \\ r_{ij} &\approx 0 + 0.1z_{ij}, i \neq j, i \geq 3, \end{aligned}$$

where $z_{ji} = -z_{ij}$ are standard normal variates. From (11), the decision rule δ_{HT} in this case is mainly on comparing $\sum_{j:j\neq 1} r_{1j}$ and $\sum_{j:j\neq 2} r_{2j}$. The difference between these two sums is $\frac{1}{2} + 0.1(\sum_{j:j\neq 1} z_{1j} - \sum_{j:j\neq 2} z_{2j})$, where the second term has zero mean and, when k is large, high variance. Therefore, for large k, the decision depends strongly on these normal variates, and the probability of choosing the first class is approaching half. On the other hand, δ_{PKPD} relies on comparing $\sum_{j:j\neq 1} 1/r_{1j}$ and $\sum_{j:j\neq 2} 1/r_{2j}$. As the difference between $1/r_{12}$ and $1/r_{21}$ is larger than that between r_{12} and r_{21}, though the accuracy rates decline when k increases, the situation is less serious.

We also analyze the mean square error (MSE) in Figure 3:

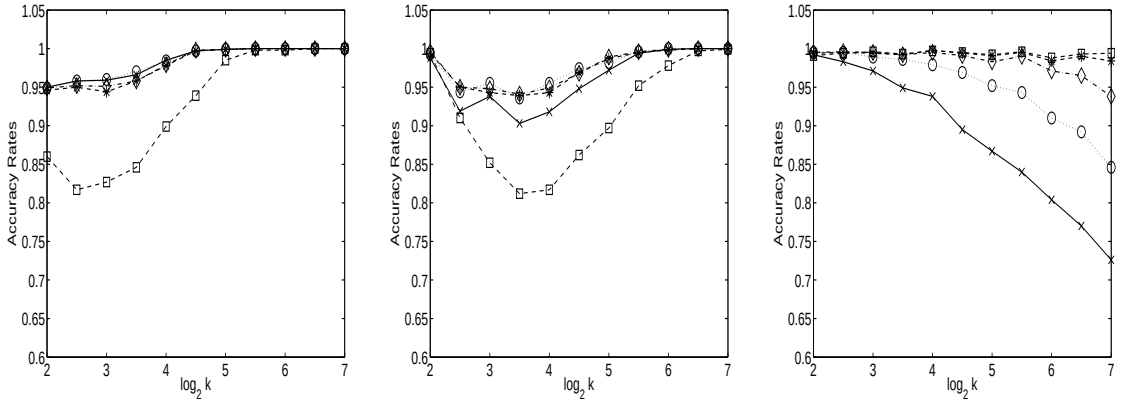
$$\text{MSE} = \frac{1}{1000} \sum_{j=1}^{1000} \frac{1}{k} \sum_{i=1}^{k} (\hat{p}_i^j - p_i)^2, \tag{35}$$

where $\hat{\mathbf{p}}^j$ is the probability estimate obtained in the jth of the 1,000 replicates. Overall, δ_{HT} and δ_V have higher MSE, confirming again that they are less stable. Note that Algorithm 1 and (11) give the same prediction for δ_{HT}, but their MSE are different. Here we consider (11) as it is the one analyzed and compared in Section 5.

In summary, δ_1 and δ_2 are less sensitive to p_i, and their overall performance are fairly stable. All observations about δ_{HT}, δ_1, δ_2, and δ_V here agree with our analysis in Section 5. Despite some similarity to δ_{HT}, δ_{PKPD} outperforms δ_{HT} in general. Experiments in this study are conducted using MATLAB.

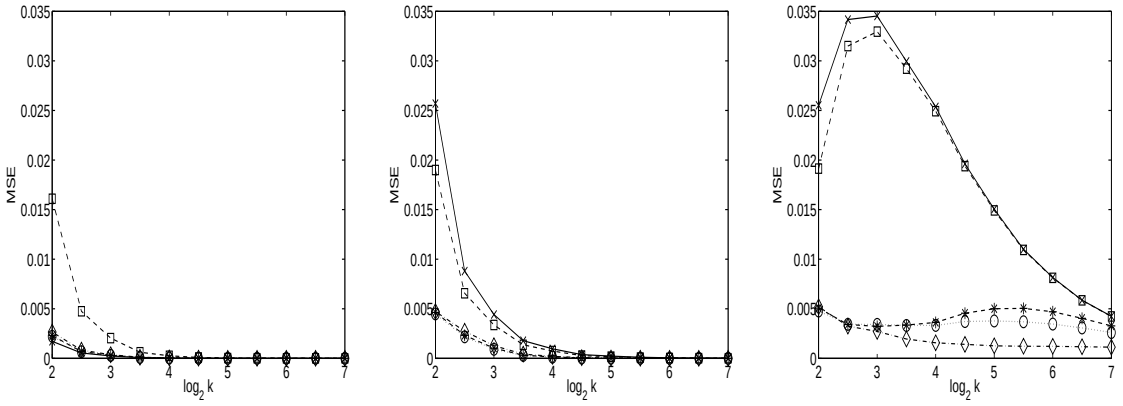
7. Experiments on Real Data

In this section we present experimental results on several multi-class problems: dna, satimage, segment, and letter from the Statlog collection (Michie et al., 1994), waveform from UCI Machine Learning Repository (Blake and Merz, 1998), USPS (Hull, 1994), and MNIST (LeCun et al., 1998). The numbers of classes and features are reported in Table 7. Except dna, which takes two possible values 0 and 1, each attribute of all other data is linearly scaled to $[-1, 1]$. In each scaled data, we randomly select 300 training and 500 testing instances from thousands of data points. 20 such selections are generated and the testing error rates are averaged. Similarly, we do experiments on larger sets (800 training and 1,000 testing). All training and testing sets used are available at `http://`



(a) (b) (c)

Figure 2: Accuracy of predicting the true class by the methods: δ_{HT} (solid line, cross marked), δ_V (dashed line, square marked), δ_1 (dotted line, circle marked), δ_2 (dashed line, asterisk marked), and δ_{PKPD} (dashdot line, diamond marked).



(a) (b) (c)

Figure 3: MSE by the methods: δ_{HT} via (11) (solid line, cross marked), δ_V (dashed line, square marked), δ_1 (dotted line, circle marked), δ_2 (dashed line, asterisk marked), and δ_{PKPD} (dashdot line, diamond marked).

`www.csie.ntu.edu.tw/~cjlin/papers/svmprob/data` and the code is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/svmprob/svmprob-1.0.tgz`.

For the implementation of the four probability estimates, δ_1 and δ_2 are via solving linear systems. For δ_{HT}, we implement Algorithm 1 with the following stopping condition

$$\sum_{i=1}^{k} \left| \frac{\sum_{j:j\neq i} r_{ij}}{\sum_{j:j\neq i} \mu_{ij}} - 1 \right| \leq 10^{-3}.$$

We observe that the performance of δ_{HT} may downgrade if the stopping condition is too loose.

dataset	dna	waveform	satimage	segment	USPS	MNIST	letter
#class	3	3	6	7	10	10	26
#attribute	180	21	36	19	256	784	16

Table 1: Data set Statistics

7.1 SVM as the Binary Classifier

We first consider support vector machines (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) with the RBF kernel $e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}$ as the binary classifier. The regularization parameter C and the kernel parameter γ are selected by cross-validation (CV). To begin, for each training set, a five-fold cross-validation is conducted on the following points of (C,γ): $[2^{-5}, 2^{-3}, \ldots, 2^{15}] \times [2^{-5}, 2^{-3}, \ldots, 2^{15}]$. This is done by modifying LIBSVM (Chang and Lin, 2001), a library for SVM. At each (C,γ), sequentially four folds are used as the training set while one fold as the validation set. The training of the four folds consists of $k(k-1)/2$ binary SVMs. For the binary SVM of the ith and jth classes, we employ an improved implementation (Lin et al., 2003) of Platt's posterior probabilities (Platt, 2000) to estimate r_{ij}:

$$r_{ij} = P(i \mid i \text{ or } j, \mathbf{x}) = \frac{1}{1+e^{A\hat{f}+B}}, \tag{36}$$

where A and B are estimated by minimizing the negative log-likelihood function, and $\hat{f}$ are the decision values of training data. Platt (2000) and Zhang (2004) observe that SVM decision values are easily clustered at ± 1, so the probability estimate (36) may be inaccurate. Thus, it is better to use CV decision values as we less overfit the model and values are not so close to ± 1. In our experiments here, this requires a further CV on the four-fold data (i.e., a second level CV).

Next, for each instance in the validation set, we apply the pairwise coupling methods to obtain classification decisions. The error of the five validation sets is thus the cross-validation error at (C,γ). From this, each rule obtains its best (C,γ).[2] Then, the decision values from the five-fold cross-validation at the best (C,γ) are employed in (36) to find the final A and B for future use. These two values and the model via applying the best parameters on the whole training set are then used to predict testing data. Figure 4 summarizes the procedure of getting validation accuracy at each given (C,γ).

2. If more than one parameter sets return the smallest cross-validation error, we simply choose the one with the smallest C.

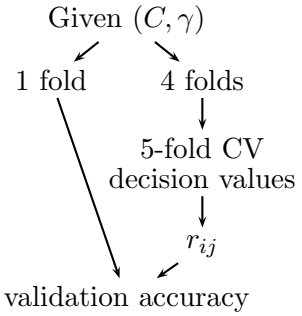


Figure 4: Parameter selection when using SVM as the binary classifier

The average of 20 MSEs are presented on the left panel of Figure 5, where the solid line represents results of small sets (300 training/500 testing), and the dashed line of large sets (800 training/1,000 testing). The definition of MSE here is similar to (35), but as there is no correct p_i for these problems, we let $p_i = 1$ if the data is in the ith class, and 0 otherwise. This measurement is called Brier Score (Brier, 1950), which is popular in meteorology. The figures show that for smaller k, δ_{HT}, δ_1, δ_2 and δ_{PKPD} have similar MSEs, but for larger k, δ_{HT} has the largest MSE. The MSEs of δ_V are much larger than those by all other methods, so they are not included in the figures. In summary, the two proposed approaches, δ_1 and δ_2, are fairly insensitive to the values of k, and all above observations agree well with previous findings in Sections 5 and 6.

Next, left panels of Figures 6 and 7 present the average of 20 test errors for problems with small size (300 training/500 testing) and large size (800 training/1,000 testing), respectively. The caption of each sub-figure also shows the average of 20 test errors of the multi-class implementation in LIBSVM. This rule is voting using merely pairwise SVM decision values, and is denoted as δ_{DV} for later discussion. The figures show that the errors of the five methods are fairly close for smaller k, but quite different for larger k. Notice that for smaller k (Figures 6 and 7 (a), (c), (e), and (g)) the differences of the averaged errors among the five methods are small, and there is no particular trend in these figures. However, for problems with larger k (Figures 6 and 7 (i), (k), and (m)), the differences are bigger and δ_{HT} is less competitive. In particular, for letter problem (Figure 6 (m), k =26), δ_2 and δ_V outperform δ_{HT} by more than 4%. The test errors along with MSE seems to indicate that, for problems with larger k, the posterior probabilities p_i are closer to the setting of Figure 2(c), rather than that of Figure 2(a). Another feature consistent with earlier findings is that when k is larger the results of δ_2 are closer to those of δ_V, and δ_1 closer to δ_{HT}, for both small and large training/testing sets. As for δ_{PKPD}, its overall performance is competitive, but we are not clear about its relationships to the other methods.

Finally, we consider another criterion on evaluating the probability estimates: the likelihood.

$$\prod_{j=1}^{l} p_{y_j}^j$$

In practice, we use its log likelihood and divide the value by a scaling factor l:

$$\frac{1}{l}\sum_{j=1}^{l}\log p^j_{y_j}, \tag{37}$$

where l is the number of test data, $\mathbf{p}^j$ is the probability estimates for the jth data, and y_j is its actual class label.

A larger value implies a possibly better estimate. The left panel of Figure 8 presents the results of using SVM as the binary classifier. Clearly the trend is the same as MSE and accuracy. When k is larger, δ_2 and δ_V have larger values and hence better performance. Similar to MSE, values of δ_V are not presented as they are too small.

7.2 Random Forest as the Binary Classifier

In this subsection we consider random forest (Breiman, 2001) as the binary classifier and conduct experiments on the same data sets. As random forest itself can provide multi-class probability estimates, we denote the corresponding rule as δ_{RF} and also compare it with the coupling methods.

For each two classes of data, we construct 500 trees as the random forest classifiers. Using m_{try} randomly selected features, a bootstrap sample (around two thirds) of training data are employed to generate a full tree without pruning. For each test instance, r_{ij} is simply the proportion out of the 500 trees that class i wins over class j. As we set the number of trees to be fixed at 500, the only parameter left for tuning is m_{try}. Similar to (Sventnik et al., 2003), we select m_{try} from $\{1, \sqrt{m}, m/3, m/2, m\}$ by five-fold cross validation, where m is the number of attributes. The cross validation procedure first sequentially uses four folds as the training set to construct $k(k-1)/2$ pairwise random forests, next obtains the decision for each instance in the validation set by the pairwise coupling methods, and then calculates the cross validation error at the given m_{try} by the error of five validation sets. This is similar to the procedure in Section 7.1, but we do not need a second-level CV for obtaining accurate two-class probabilistic estimates (i.e., r_{ij}). Instead of CV, a more efficient "out of bag" validation can be used for random forest, but here we keep using CV for consistency. Experiments are conducted using an R-interface (Liaw and Wiener, 2002) to the code from (Breiman, 2001).

The MSE presented in the right panel of Figure 5 shows that δ_1 and δ_2 yield more stable results than δ_{HT} and δ_V for both small and large sets. The right panels of Figures 6 and 7 give the average of 20 test errors. The caption of each sub-figure also shows the averaged error when using random forest as a multi-class classifier (δ_{RF}). Notice that random forest bears a resemblance to SVM: the errors are only slightly different among the five methods for smaller k, but δ_V and δ_2 tend to outperform δ_{HT} and δ_1 for larger k. The right panel of Figure 8 presents the log likelihood value (37). The trend is again the same. In summary, the results by using random forest as the binary classifier strongly support previous findings regarding the four methods.

7.3 Miscellaneous Observations and Discussion

Recall that in Section 7.1 we consider δ_{DV}, which does not use Platt's posterior probabilities. Experimental results in Figure 6 show that δ_{DV} is quite competitive (in particular, 3% better for letter), but is about 2% worse than all probability-based methods for waveform. Similar observations on waveform are also reported in (Duan and Keerthi, 2003), where the comparison is between δ_{DV} and δ_{HT}. We explain why the results by probability-based and decision-value-based methods can be so distinct. For some problems, the parameters selected by δ_{DV} are quite different from those by the other five rules. In waveform, at some parameters all probability-based methods gives much higher cross validation accuracy than δ_{DV}. We observe, for example, the decision values of validation sets are in [0.73, 0.97] and [0.93, 1.02] for data in two classes; hence, all data in the validation sets are classified as in one class and the error is high. On the contrary, the probability-based methods fit the decision values by a sigmoid function, which can better separate the two classes by cutting at a decision value around 0.95. This observation shed some light on the difference between probability-based and decision-value based methods.

Though the main purpose of this section is to compare different probability estimates, here we check the accuracy of another multi-class classification method: exponential loss-based decoding by Allwein et al. (2001). In the pairwise setting, if $\hat{f}_{ij} \in R$ is the two-class hypothesis so that $\hat{f}_{ij} > 0$ (< 0) predicts the data to be in the ith (jth) class, then

$$\text{predicted label} = \arg\min_i \left(\sum_{j:j<i} e^{\hat{f}_{ji}} + \sum_{j:j>i} e^{-\hat{f}_{ij}} \right). \tag{38}$$

For SVM, we can simply use decision values as $\hat{f}_{ij}$. On the other hand, $r_{ij} - 1/2$ is another choice. Table 2 presents the error of the seven problem using these two options. Results indicate that using decision values is worse than $r_{ij} - 1/2$ when k is large (USPS,MNIST, and letter). This observation seems to indicate that large numerical ranges of $\hat{f}_{ij}$ may cause (38) to have more erroneous results ($r_{ij} - 1/2$ is always in $[-1/2, 1/2]$). The results of using $r_{ij} - 1/2$ is competitive with those in Figures 6 and 7 when k is small. However, for larger k (e.g., letter), it is slightly worse than δ_2 and δ_V. We think this result is due to the similarity between (38) and δ_{HT}. When $\hat{f}_{ij}$ is close to zero, $e^{\hat{f}_{ij}} \approx 1 + \hat{f}_{ij}$, so (38) reduces to a "linear loss-based encoding." When $r_{ij} - 1/2$ is used, $\hat{f}_{ji} = r_{ji} - 1/2 = 1/2 - r_{ij}$. Thus, the linear encoding is $\arg\min_i[\sum_{j:j\neq i} -r_{ij}] \equiv \arg\max_i[\sum_{j:j\neq i} r_{ij}]$, exactly the same as (11) of δ_{HT}.

training/testing ($\hat{f}_{ij}$)	dna	waveform	satimage	segment	USPS	MNIST	letter
300/500 (dec. values)	10.47	16.23	14.12	6.21	11.57	14.99	38.59
300/500 ($r_{ij} - 1/2$)	10.47	15.11	14.45	6.03	11.08	13.58	38.27
800/1000 (dec. values)	6.36	14.20	11.55	3.35	8.47	8.97	22.54
800/1000 ($r_{ij} - 1/2$)	6.22	13.45	11.6	3.19	7.71	7.95	20.29

Table 2: Average of 20 test errors using exponential loss-based decoding (in percentage)

Regarding the accuracy of pairwise (i.e., δ_{DV}) and non-pairwise (e.g., "one-against-the-rest") multi-class classification methods, there are already excellent comparisons. As δ_V

and δ_2 have similar accuracy to δ_{DV}, roughly how non-pairwise methods compared to δ_{DV} is the same as compared to δ_V and δ_2.

The results of random forest as a multi-class classifier (i.e., δ_{RF}) are reported in the caption of each sub-figure in Figures 6 and 7. We observe from the figures that, when the number of classes is larger, using random forest as a multi-class classifier is better than coupling binary random forests. However, for dna ($k = 3$) the result is the other way around. This observation for random forest is left as a future research issue.

Acknowledgments

The authors thank S. Sathiya Keerthi for helpful comments. They also thank the associate editor and anonymous referees for useful comments. This work was supported in part by the National Science Council of Taiwan via the grants NSC 90-2213-E-002-111 and NSC 91-2118-M-004-003.

Appendix A. Proof of Theorem 2

It suffices to prove that any optimal solution $\mathbf{p}$ of (20) satisfies $p_i \geq 0, i = 1, \ldots, k$. If this is not true, without loss of generality, we assume

$$p_1 < 0, \ldots, p_r < 0, p_{r+1} \geq 0, \ldots, p_k \geq 0,$$

where $r < k$ because $\sum_{i=1}^{k} p_i = 1$. We can then define a new feasible solution of (20):

$$p'_1 = 0, \ldots, p'_r = 0, p'_{r+1} = p_{r+1}/\alpha, \ldots, p'_k = p_k/\alpha,$$

where $\alpha = 1 - \sum_{i=1}^{r} p_i > 1$.

With $r_{ij} > 0$ and $r_{ji} > 0$, we obtain

$$\begin{aligned}
&(r_{ji}p_i - r_{ij}p_j)^2 \geq 0 = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ if } 1 \leq i, j \leq r,\\
&(r_{ji}p_i - r_{ij}p_j)^2 > \frac{(r_{ij}p_j)^2}{\alpha^2} = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ if } 1 \leq i \leq r, r+1 \leq j \leq k,\\
&(r_{ji}p_i - r_{ij}p_j)^2 \geq \frac{(r_{ji}p_i - r_{ij}p_j)^2}{\alpha^2} = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ if } r+1 \leq i, j \leq k.
\end{aligned}$$

Therefore,

$$\sum_{i=1}^{k} \sum_{j:j\neq i} (r_{ij}p_i - r_{ji}p_j)^2 > \sum_{i=1}^{k} \sum_{j:j\neq i} (r_{ij}p'_i - r_{ji}p'_j)^2.$$

This contradicts the assumption that $\mathbf{p}$ is an optimal solution of (20).

Appendix B. Proof of Theorem 3

(i) If $Q + \Delta\mathbf{e}\mathbf{e}^T$ is not positive definite, there is a vector $\mathbf{v}$ with $v_i \neq 0$ such that

$$\mathbf{v}^T(Q + \Delta\mathbf{e}\mathbf{e}^T)\mathbf{v} = \frac{1}{2}\sum_{t=1}^{k} \sum_{u:u\neq t} (r_{ut}v_t - r_{tu}v_u)^2 + \Delta(\sum_{t=1}^{k} v_t)^2 = 0. \tag{39}$$

For all $t \neq i$, $r_{it}v_t - r_{ti}v_i = 0$, so

$$v_t = \frac{r_{ti}}{r_{it}} v_i \neq 0.$$

Thus,

$$\sum_{t=1}^{k} v_t = (1 + \sum_{t:t\neq i} \frac{r_{ti}}{r_{it}}) v_i \neq 0,$$

which contradicts (39).

The positive definiteness of $Q + \Delta \mathbf{e}\mathbf{e}^T$ implies that $\begin{bmatrix} Q+\Delta \mathbf{e}\mathbf{e}^T & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is invertible. As $\begin{bmatrix} Q+\Delta \mathbf{e}\mathbf{e}^T & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is from adding the last row of $\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ to its first k rows (with a scaling factor Δ), the two matrices have the same rank. Thus, $\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is invertible as well. Then (23) has a unique solution, and so does (19).

(ii) If Q is not positive definite, there is a vector $\mathbf{v}$ with $v_i \neq 0$ such that

$$\mathbf{v}^T Q \mathbf{v} = \frac{1}{2} \sum_{t=1}^{k} \sum_{u:u\neq t} (r_{ut}v_t - r_{tu}v_u)^2 = 0.$$

Therefore,

$$(r_{ut}v_t - r_{tu}v_u)^2 = 0, \forall t \neq u.$$

As $r_{tu} > 0, \forall t \neq u$, for any $s \neq j$ for which $s \neq i$ and $j \neq i$, we have

$$v_s = \frac{r_{si}}{r_{is}} v_i, \qquad v_j = \frac{r_{ji}}{r_{ij}} v_i, \qquad v_s = \frac{r_{sj}}{r_{js}} v_j. \tag{40}$$

As $v_i \neq 0$, (40) implies

$$\frac{r_{si}r_{sj}}{r_{is}} = \frac{r_{ji}r_{js}}{r_{ij}},$$

which contradicts (25).

Appendix C. Proof of Theorem 4

First we need a lemma to show the strict decrease of the objective function:

Lemma 5 *If $r_{ij} > 0, \forall i \neq j$, $\mathbf{p}$ and $\mathbf{p}^n$ are from two consecutive iterations of Algorithm 2, and $\mathbf{p}^n \neq \mathbf{p}$, then*

$$\frac{1}{2}(\mathbf{p}^n)^T Q \mathbf{p}^n < \frac{1}{2}\mathbf{p}^T Q \mathbf{p}. \tag{41}$$

Proof. Assume that p_t is the component to be updated. Then, $\mathbf{p}^n$ is obtained through the following calculation:

$$\bar{p}_i = \begin{cases} p_i & \text{if } i \neq t, \\ \frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_j + \mathbf{p}^T Q \mathbf{p}) & \text{if } i = t, \end{cases} \tag{42}$$

and

$$\mathbf{p}^n = \frac{\bar{\mathbf{p}}}{\sum_{i=1}^k \bar{p}_i}. \tag{43}$$

For (43) to be a valid operation, $\sum_{i=1}^k \bar{p}_i$ must be strictly positive. To show this, we first suppose that the current solution $\mathbf{p}$ satisfies $p_i \geq 0, i = 1, \ldots, l$, but the next solution $\bar{\mathbf{p}}$ has $\sum_{i=1}^k \bar{p}_i = 0$. In Section 4.2, we have shown that $\bar{p}_t \geq 0$, so with $\bar{p}_i = p_i \geq 0, \forall i \neq t$, $\bar{p}_i = 0$ for all i. Next, from (42), $p_i = \bar{p}_i = 0$ for $i \neq t$, which, together with the equality $\sum_{i=1}^k p_i = 1$ implies that $p_t = 1$. However, if $p_t = 1$ and $p_i = 0$ for $i \neq t$, then $\bar{p}_t = 1$ from (42). This contradicts the situation that $\bar{p}_i = 0$ for all i. Therefore, by induction, the only requirement is to have nonnegative initial $\mathbf{p}$.

To prove (41), first we rewrite the update rule (42) as

$$\begin{aligned} \bar{p}_t &= p_t + \frac{1}{Q_{tt}}(-(Q\mathbf{p})_t + \mathbf{p}^T Q\mathbf{p}) \\ &= p_t + \Delta. \end{aligned} \tag{44}$$

Since we keep $\sum_{i=1}^k p_i = 1$, $\sum_{i=1}^k \bar{p}_i = 1 + \Delta$. Then

$$\begin{aligned} & \bar{\mathbf{p}}^T Q\bar{\mathbf{p}} - (\sum_{i=1}^k \bar{p}_i)^2 \mathbf{p}^T Q\mathbf{p} \\ = \; & \mathbf{p}^T Q\mathbf{p} + 2\Delta(Q\mathbf{p})_t + Q_{tt}\Delta^2 - (1+\Delta)^2 \mathbf{p}^T Q\mathbf{p} \\ = \; & 2\Delta(Q\mathbf{p})_t + Q_{tt}\Delta^2 - (2\Delta + \Delta^2)\mathbf{p}^T Q\mathbf{p} \\ = \; & \Delta\left(2(Q\mathbf{p})_t - 2\mathbf{p}^T Q\mathbf{p} + Q_{tt}\Delta - \Delta\mathbf{p}^T Q\mathbf{p}\right) \\ = \; & \Delta(-Q_{tt}\Delta - \Delta\mathbf{p}^T Q\mathbf{p}) \qquad (45) \\ = \; & -\Delta^2(Q_{tt} + \mathbf{p}^T Q\mathbf{p}) < 0. \qquad (46) \end{aligned}$$

(45) follows from the definition of Δ in (44). For (46), it uses $Q_{tt} = \sum_{j:j\neq t} r_{jt}^2 > 0$ and $\Delta \neq 0$, which comes from the assumption $\mathbf{p}^n \neq \mathbf{p}$. □

Now we are ready to prove the theorem. If this result does not hold, there is a convergent sub-sequence $\{\mathbf{p}^i\}_{i\in K}$ such that $\mathbf{p}^* = \lim_{i\in K, i\to\infty} \mathbf{p}^i$ is not optimal for (19). Note that there is at least one index of $\{1, \ldots, k\}$ which is considered in infinitely many iterations. Without loss of generality, we assume that for all $\mathbf{p}^i, i \in K$, p_t^i is updated to generate the next iteration $\mathbf{p}^{i+1}$. As $\mathbf{p}^*$ is not optimal for (19), starting from $t, t+1, \ldots, k, 1, \ldots, t-1$, there is a first component $\bar{t}$ for which

$$\sum_{j=1}^k Q_{\bar{t}j} p_j^* - (\mathbf{p}^*)^T Q\mathbf{p}^* \neq 0.$$

By applying one iteration of Algorithm 2 on $p_{\bar{t}}^*$, from an explanation similar to the proof of Lemma 5, we obtain $\mathbf{p}^{*,n}$ satisfying $p_{\bar{t}}^{*,n} \neq p_{\bar{t}}^*$. Then by Lemma 5,

$$\frac{1}{2}(\mathbf{p}^{*,n})^T Q\mathbf{p}^{*,n} < \frac{1}{2}(\mathbf{p}^*)^T Q\mathbf{p}^*.$$

Assume it takes $\bar{i}$ steps from t to $\bar{t}$ and $\bar{i} > 1$,

$$\begin{aligned}\lim_{i\in K, i\to\infty} p_t^{i+1} &= \lim_{i\in K, i\to\infty} \frac{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^i + (\mathbf{p}^i)^T Q\mathbf{p}^i)}{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^i + (\mathbf{p}^i)^T Q\mathbf{p}^i) + \sum_{j:j\neq t} p_j^i} \\ &= \frac{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^* + (\mathbf{p}^*)^T Q\mathbf{p}^*)}{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^* + (\mathbf{p}^*)^T Q\mathbf{p}^*) + \sum_{j:j\neq t} p_j^*} \\ &= \frac{p_t^*}{\sum_{j=1}^k p_j^*} = p_t^*,\end{aligned}$$

we have

$$\lim_{i\in K, i\to\infty} \mathbf{p}^i = \lim_{i\in K, i\to\infty} \mathbf{p}^{i+1} = \cdots = \lim_{i\in K, i\to\infty} \mathbf{p}^{i+\bar{i}-1} = \mathbf{p}^*.$$

Moreover,

$$\lim_{i\in K, i\to\infty} \mathbf{p}^{i+\bar{i}} = \mathbf{p}^{*,n}$$

and

$$\begin{aligned}\lim_{i\in K, i\to\infty} \frac{1}{2}(\mathbf{p}^{i+\bar{i}})^T Q\mathbf{p}^{i+\bar{i}} &= \frac{1}{2}(\mathbf{p}^{*,n})^T Q\mathbf{p}^{*,n} \\ &< \frac{1}{2}(\mathbf{p}^*)^T Q\mathbf{p}^* \\ &= \lim_{i\in K, i\to\infty} \frac{1}{2}(\mathbf{p}^i)^T Q\mathbf{p}^i.\end{aligned}$$

This contradicts the fact from Lemma 5:

$$\frac{1}{2}(\mathbf{p}^1)^T Q\mathbf{p}^1 \geq \frac{1}{2}(\mathbf{p}^2)^T Q\mathbf{p}^2 \geq \cdots \geq \frac{1}{2}(\mathbf{p}^*)^T Q\mathbf{p}^*.$$

Therefore, $\mathbf{p}^*$ must be optimal for (19).

Appendix D. Implementation Notes of Algorithm 2

From Algorithm 2, the main operation of each iteration is on calculating $-\sum_{j:j\neq t} Q_{tj}p_j$ and $\mathbf{p}^T Q\mathbf{p}$, both $O(k^2)$ procedures. In the following, we show how to easily reduce the cost per iteration to $O(k)$.

Following the notation in Lemma 5 of Appendix D, we consider $\mathbf{p}$ the current solution. Assume p_t is the component to be updated, we generate $\bar{\mathbf{p}}$ according to (42) and normalize $\bar{\mathbf{p}}$ to the next iterate $\mathbf{p}^n$. Note that $\bar{\mathbf{p}}$ is the same as $\mathbf{p}$ except the tth component and we consider the form (44). Since $\sum_{i=1}^k p_i = 1$, (43) is $\mathbf{p}^n = \bar{\mathbf{p}}/(1+\Delta)$. Throughout iterations, we keep the current $Q\mathbf{p}$ and $\mathbf{p}^T Q\mathbf{p}$, so Δ can be easily calculated. To obtain $Q\mathbf{p}^n$ and $(\mathbf{p}^n)^T Q\mathbf{p}^n$, we use

$$\begin{aligned}(Q\mathbf{p}^n)_j &= \frac{(Q\bar{\mathbf{p}})_j}{1+\Delta} \\ &= \frac{(Q\mathbf{p})_j + Q_{jt}\Delta}{1+\Delta}, j = 1, \ldots, k, \end{aligned} \tag{47}$$

and

$$(\mathbf{p}^n)^T Q(\mathbf{p}^n) = \frac{\bar{\mathbf{p}}^T Q \bar{\mathbf{p}}}{(1+\Delta)^2} = \frac{\mathbf{p}^T Q \mathbf{p} + 2\Delta \sum_{j=1}^k (Q\mathbf{p})_j + Q_{tt}\Delta^2}{(1+\Delta)^2}. \quad (48)$$

Both (and hence the whole iteration) takes $O(k)$ operations.

Numerical inaccuracy may accumulate through iterations, so gradually (47) and (48) may be away from values directly calculated using $\mathbf{p}$. An easy prevention of this problem is to recalculate $Q\mathbf{p}$ and $\mathbf{p}^T Q\mathbf{p}$ directly using $\mathbf{p}$ after several iterations (e.g., k iterations). Then, the average cost per iteration is still $O(k) + O(k^2)/k = O(k)$.

Appendix E. Derivation of (32)

$$\begin{aligned} & \sum_{i=1}^k \sum_{j:j\neq i} (I_{\{r_{ij}>r_{ji}\}} p_j - I_{\{r_{ji}>r_{ij}\}} p_i)^2 \\ = & \sum_{i=1}^k \sum_{j:j\neq i} (I_{\{r_{ij}>r_{ji}\}} p_j^2 + I_{\{r_{ji}>r_{ij}\}} p_i^2) \\ = & 2\sum_{i=1}^k (\sum_{j:j\neq i} I_{\{r_{ji}>r_{ij}\}}) p_i^2. \end{aligned}$$

If $\sum_{j:j\neq i} I_{\{r_{ji}>r_{ij}\}} \neq 0, \forall i$, then, under the constraint $\sum_{i=1}^k p_i = 1$, the optimal solution satisfies

$$\frac{p_1}{\sum_{j:j\neq 1} I_{\{r_{j1}>r_{1j}\}}} = \cdots = \frac{p_k}{\sum_{j:j\neq k} I_{\{r_{jk}>r_{kj}\}}}.$$

Thus, (32) is the optimal solution of (31).

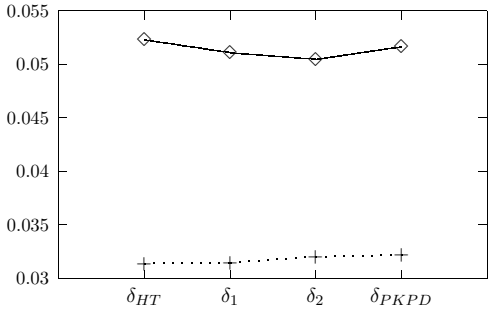
References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001. ISSN 1533-7928.

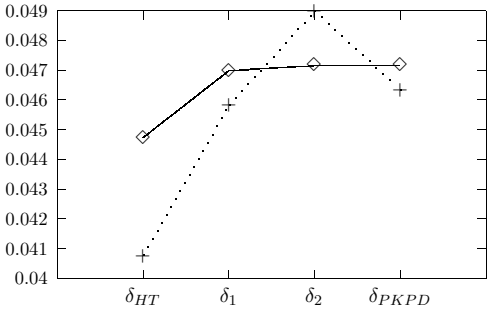
C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA, 1998. Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

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*, 1992.

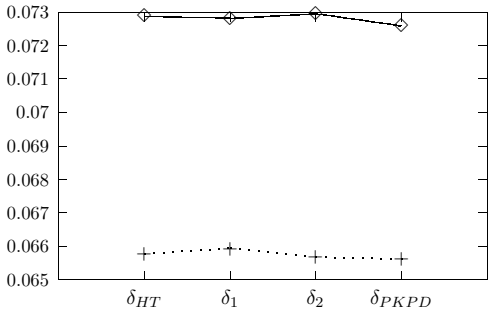
L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. URL `citeseer.nj.nec.com/breiman01random.html`.



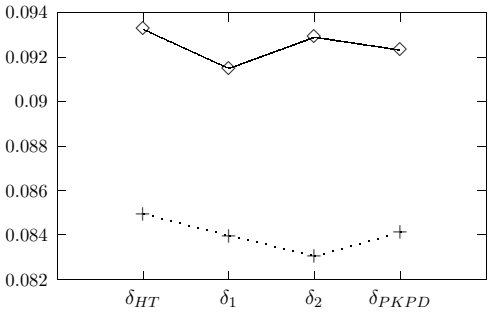
(a) dna ($k = 3$) by binary SVMs



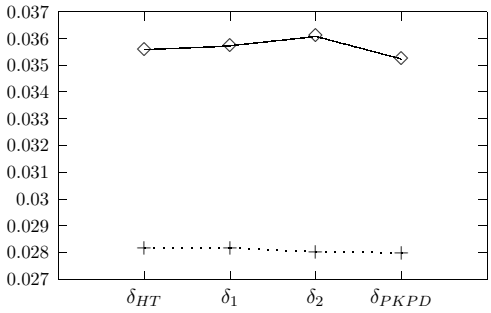
(b) dna ($k = 3$) by binary random forests



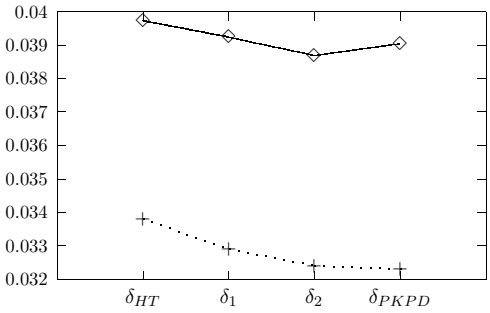
(c) waveform ($k = 3$) by binary SVMs



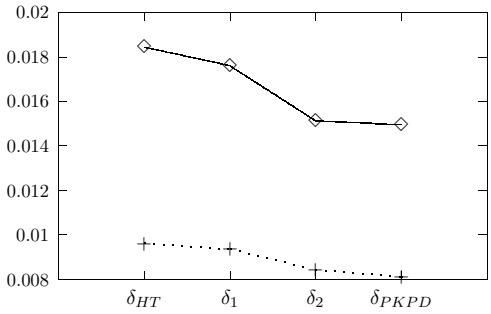
(d) waveform ($k = 3$) by binary random forests



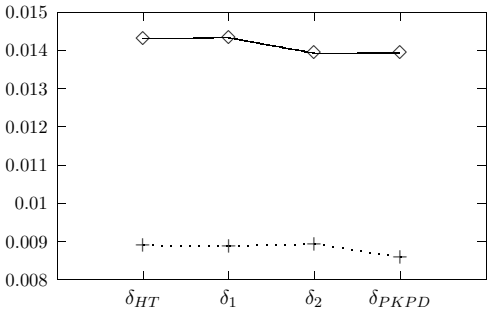
(e) satimage ($k = 6$) by binary SVMs



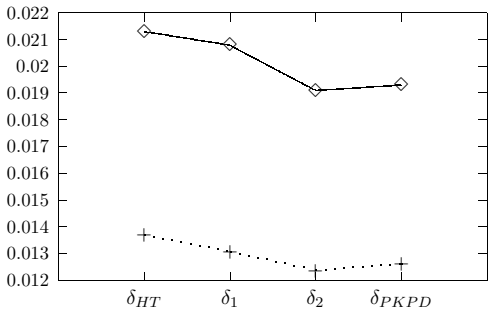
(f) satimage ($k = 6$) by binary random forests



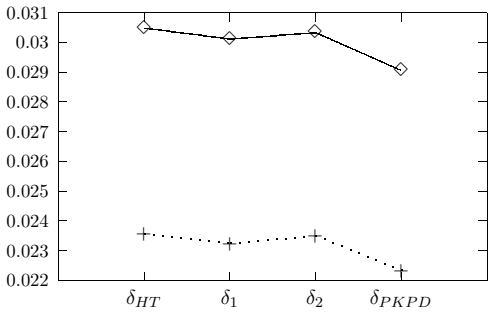
(g) segment ($k = 7$) by binary SVMs



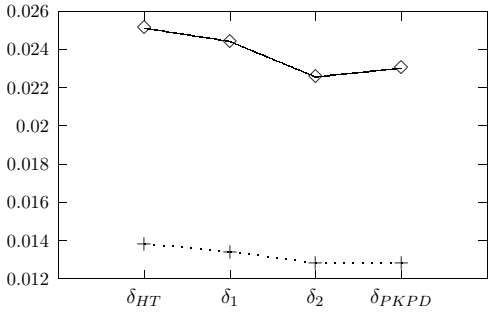
(h) segment ($k = 7$) by binary random forests



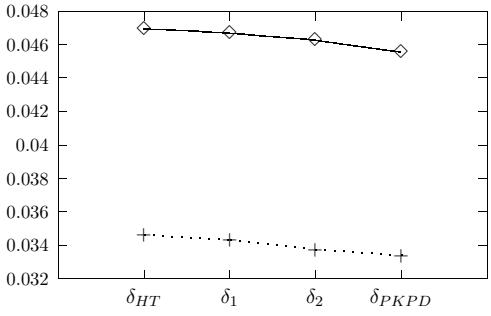
(i) USPS ($k = 10$) by binary SVMs



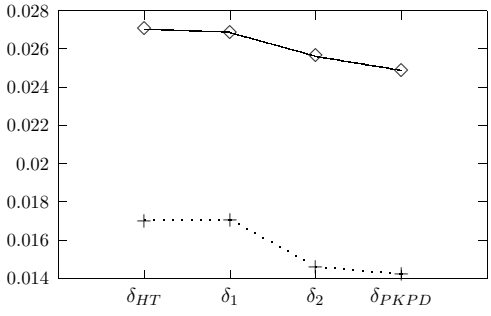
(j) USPS ($k = 10$) by binary random forests



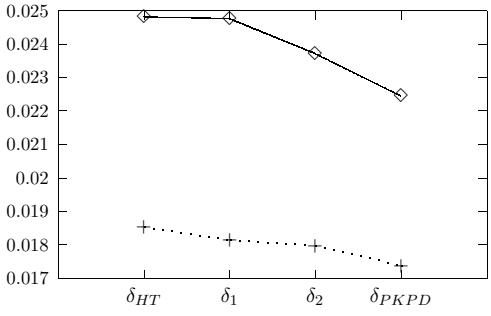
(k) MNIST ($k = 10$) by binary SVMs



(l) MNIST ($k = 10$) by binary random forests

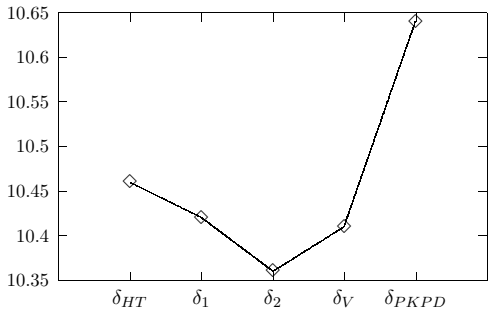


(m) letter ($k = 26$) by binary SVMs

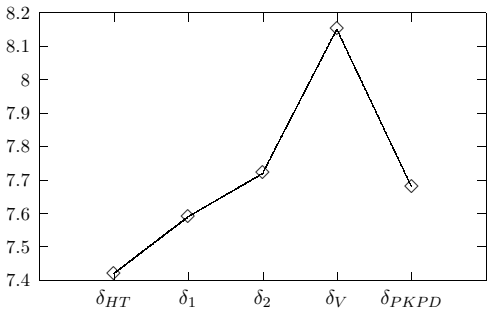


(n) letter ($k = 26$) by binary random forests

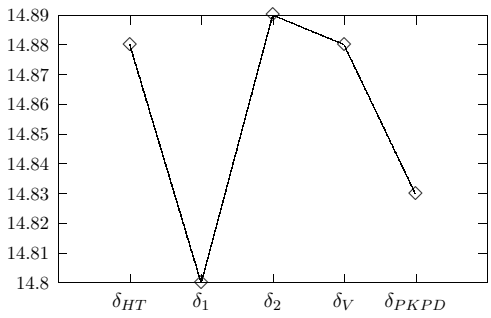
Figure 5: MSE by using four probability estimates methods based on binary SVMs (left) and binary random forests (right). MSE of δ_V is too large and is not presented. solid line: 300 training/500 testing points; dotted line: 800 training/1,000 testing points.



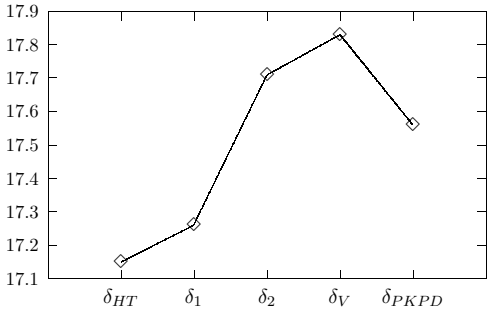
(a) dna ($k = 3$) by binary SVMs; $\delta_{DV} = 10.82\%$



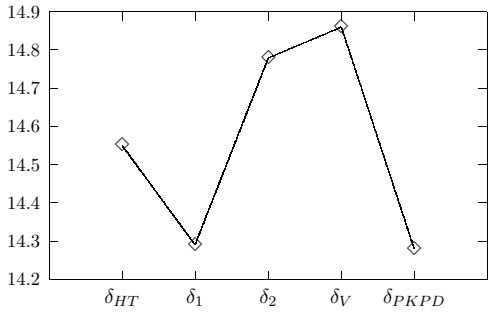
(b) dna ($k = 3$) by binary random forests; $\delta_{RF} = 8.74\%$



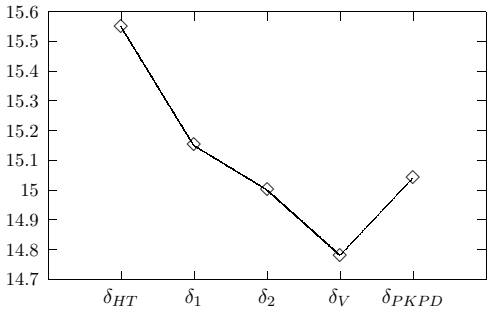
(c) waveform ($k = 3$) by binary SVMs; $\delta_{DV} = 16.47\%$



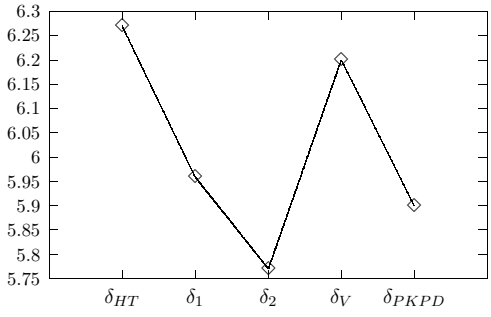
(d) waveform ($k = 3$) by binary random forests; $\delta_{RF} = 17.39\%$



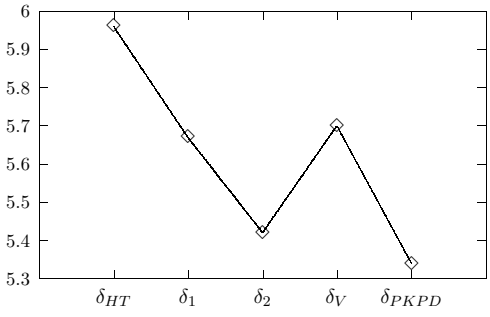
(e) satimage ($k = 6$) by binary SVMs; $\delta_{DV} = 14.88\%$



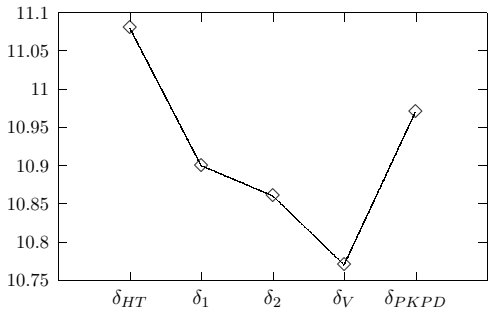
(f) satimage ($k = 6$) by binary random forests;$\delta_{RF} = 14.74\%$



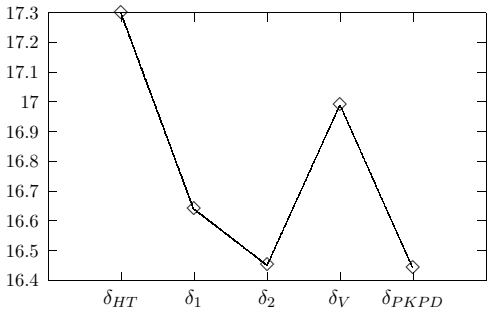
(g) segment ($k = 7$) by binary SVMs; $\delta_{DV} = 5.82\%$



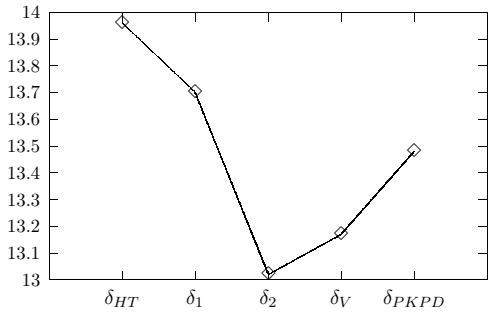
(h) segment ($k = 7$) by binary random forests; $\delta_{RF} = 5.23\%$



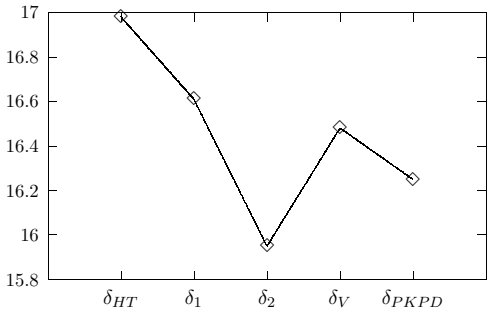
(i) USPS ($k = 10$) by binary SVMs; $\delta_{DV} = 10.71\%$



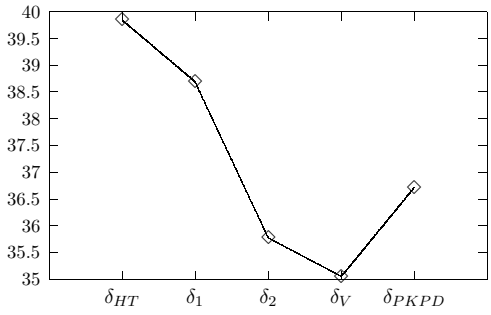
(j) USPS ($k = 10$) by binary random forests; $\delta_{RF} = 14.28\%$



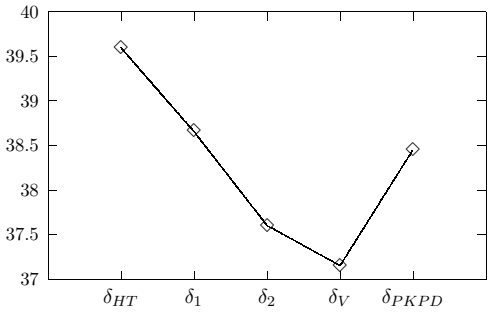
(k) MNIST ($k = 10$) by binary SVMs; $\delta_{DV} = 13.02\%$



(l) MNIST ($k = 10$) by binary random forests; $\delta_{RF} = 16.18\%$

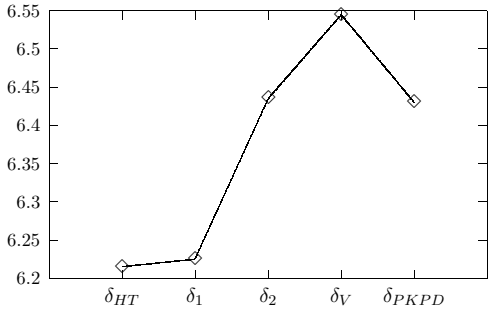


(m) letter ($k = 26$) by binary SVMs; $\delta_{DV} = 32.29\%$

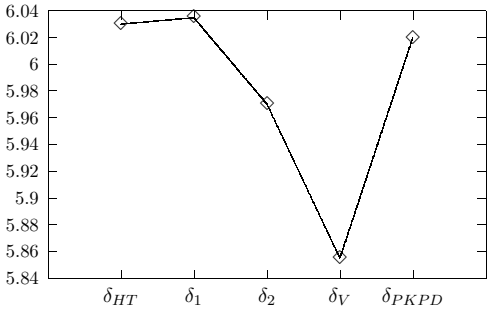


(n) letter ($k = 26$) by binary random forests; $\delta_{RF} = 32.55\%$

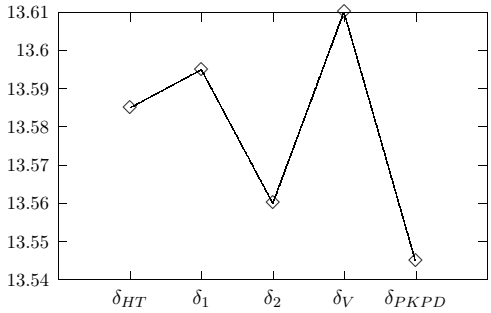
Figure 6: Average of 20 test errors by five probability estimates methods based on binary SVMs (left) and binary random forests (right). Each of the 20 test errors is by 300 training/500 testing points. Caption of each sub-figure shows the averaged error by voting using pairwise SVM decision values (δ_{DV}) and the multi-class random forest (δ_{RF}).



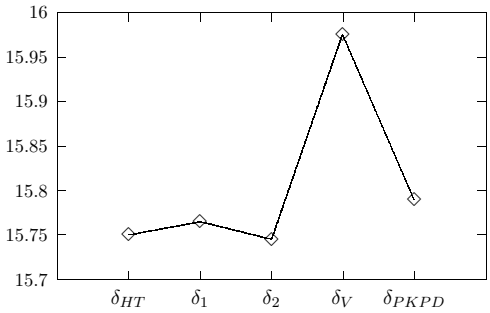
(a) dna ($k = 3$) by binary SVMs; $\delta_{DV} = 6.31\%$



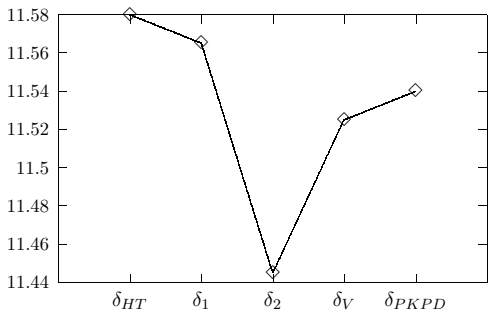
(b) dna ($k = 3$) by binary random forests; $\delta_{RF} = 6.91\%$



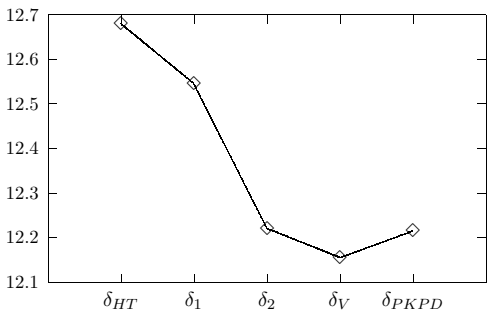
(c) waveform ($k = 3$) by binary SVMs; $\delta_{DV} = 14.33\%$



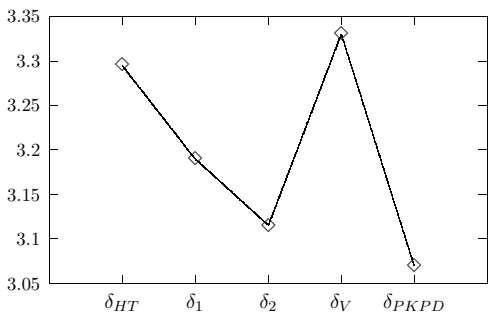
(d) waveform ($k = 3$) by binary random forests; $\delta_{RF} = 15.66\%$



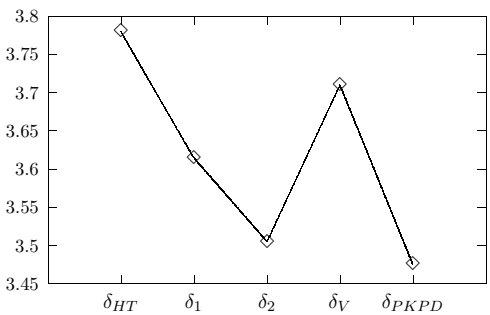
(e) satimage ($k = 6$) by binary SVMs; $\delta_{DV} = 11.54\%$



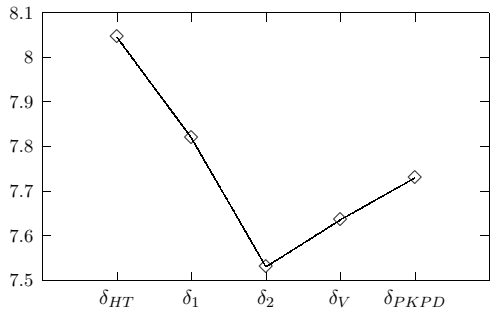
(f) satimage ($k = 6$) by binary random forests; $\delta_{RF} = 11.92\%$



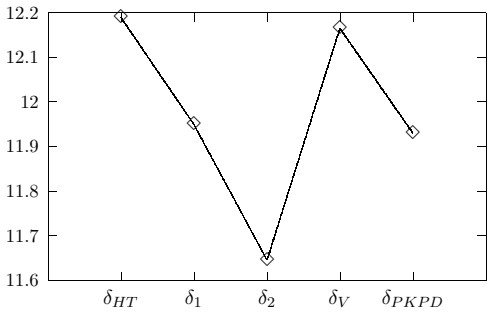
(g) segment ($k = 7$) by binary SVMs; $\delta_{DV} = 3.30\%$



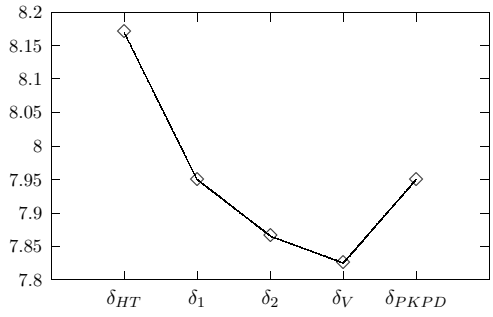
(h) segment ($k = 7$) by binary random forests; $\delta_{RF} = 2.77\%$



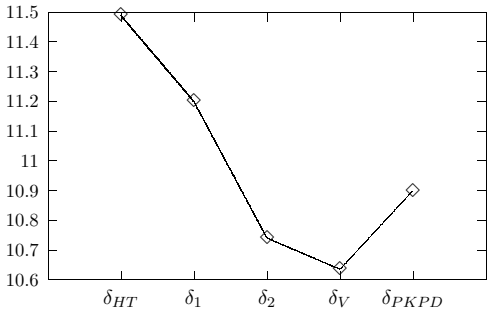
(i) USPS ($k = 10$) by binary SVMs; $\delta_{DV} = 7.54\%$



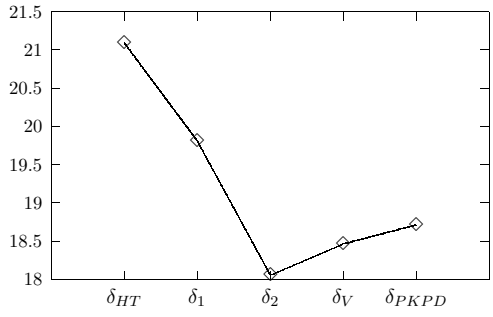
(j) USPS ($k = 10$) by binary random forests; $\delta_{RF} = 10.23\%$



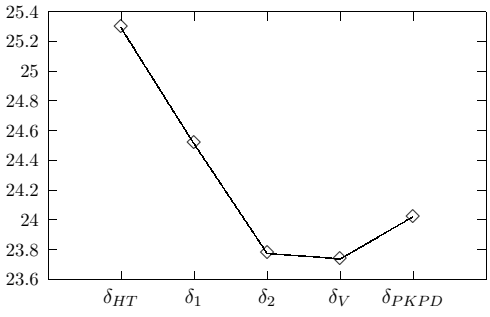
(k) MNIST ($k = 10$) by binary SVMs; $\delta_{DV} = 7.77\%$



(l) MNIST ($k = 10$) by binary random forests; $\delta_{RF} = 10.10\%$

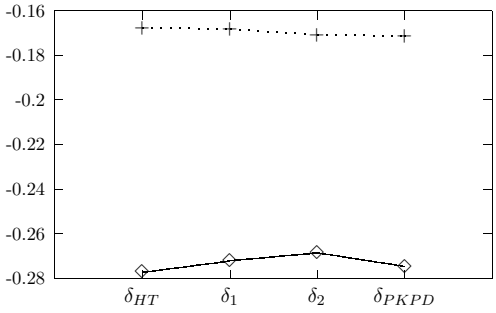


(m) letter ($k = 26$) by binary SVMs; $\delta_{DV} = 18.61\%$

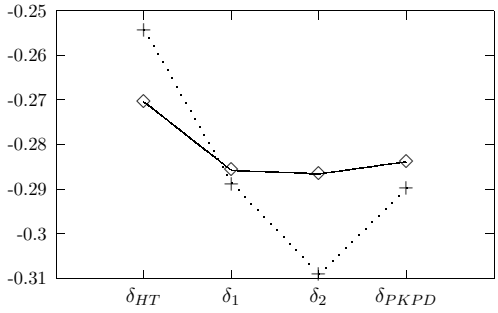


(n) letter ($k = 26$) by binary random forests; $\delta_{RF} = 20.25\%$

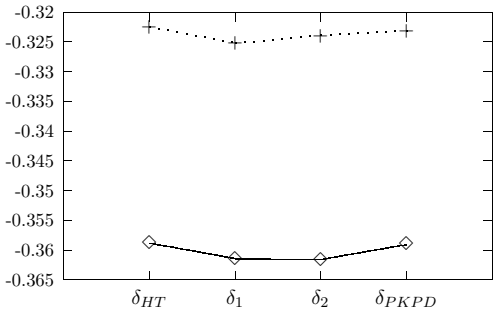
Figure 7: Average of 20 test errors by five probability estimates methods based on binary SVMs (left) and binary random forests (right). Each of the 20 test errors is by 800 training/1,000 testing points. Caption of each sub-figure shows the averaged error by voting using pairwise SVM decision values (δ_{DV}) and the multi-class random forest (δ_{RF}).



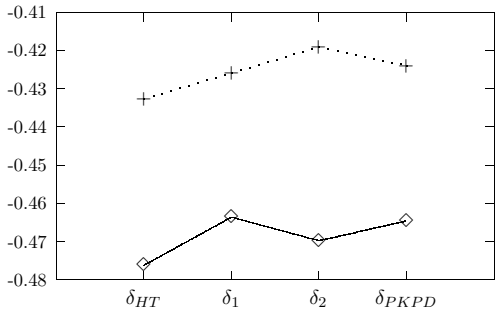
(a) dna ($k = 3$) by binary SVMs



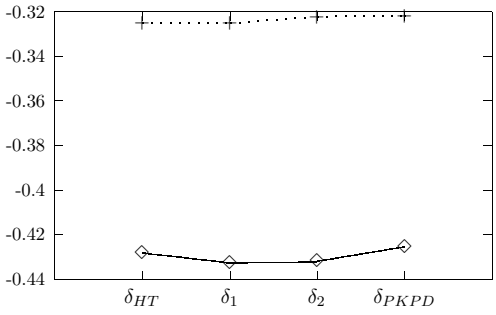
(b) dna ($k = 3$) by binary random forests



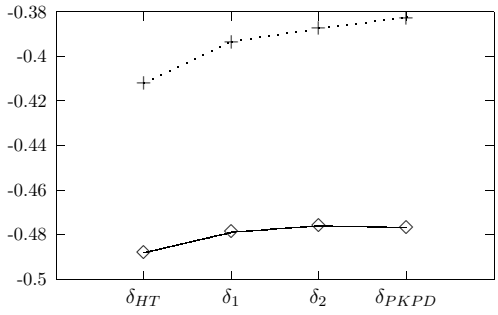
(c) waveform ($k = 3$) by binary SVMs



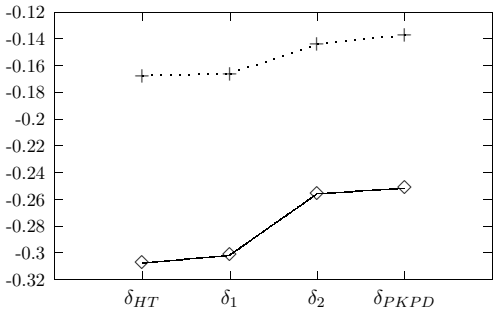
(d) waveform ($k = 3$) by binary random forests



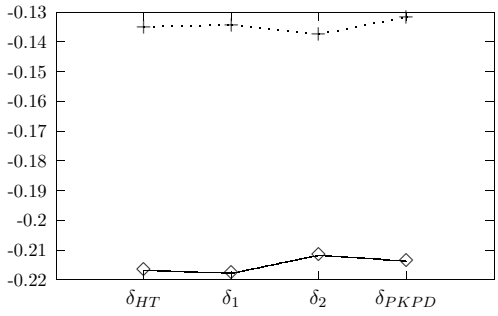
(e) satimage ($k = 6$) by binary SVMs



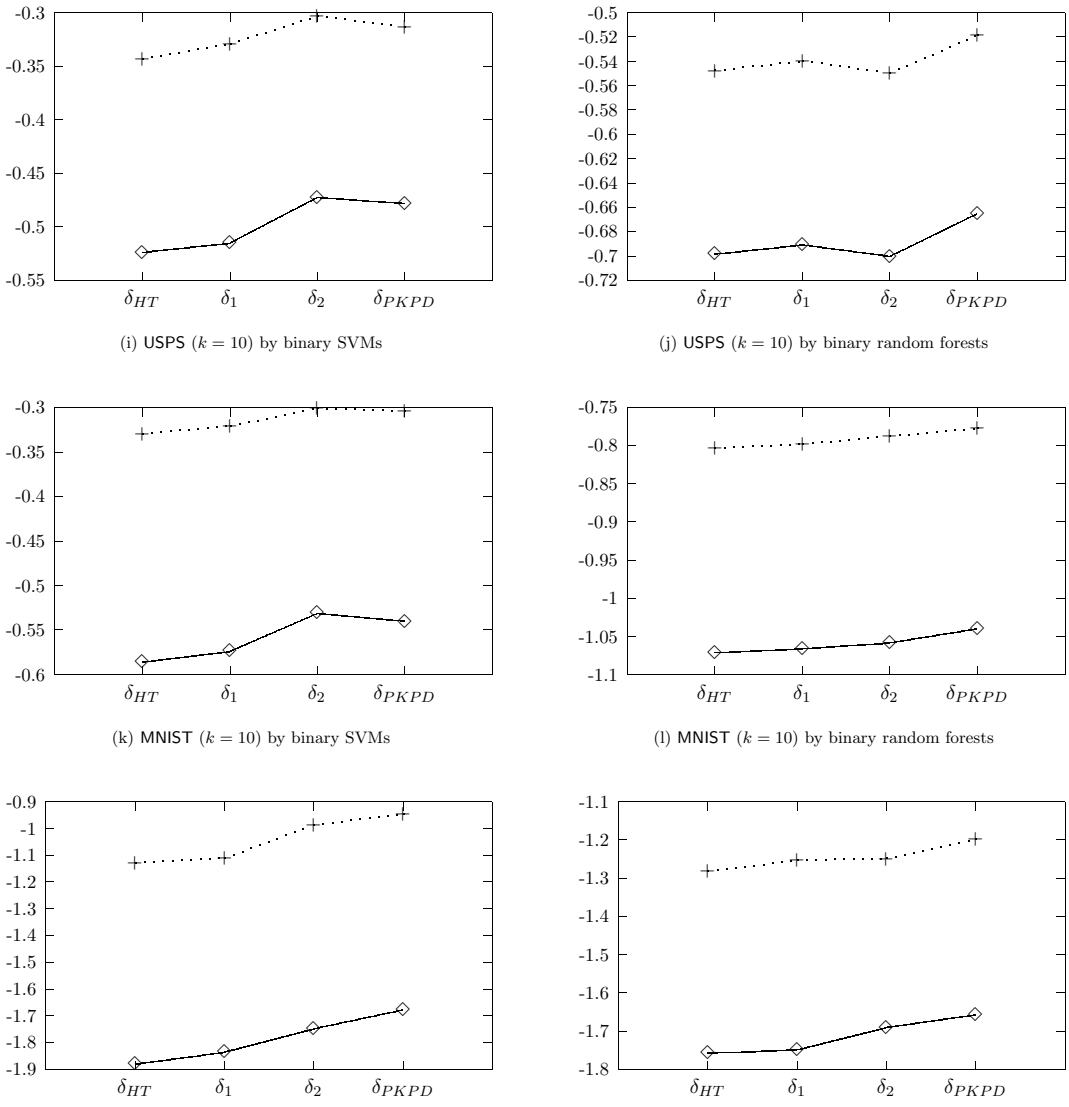
(f) satimage ($k = 6$) by binary random forests



(g) segment ($k = 7$) by binary SVMs



(h) segment ($k = 7$) by binary random forests



(i) USPS ($k = 10$) by binary SVMs

(j) USPS ($k = 10$) by binary random forests

(k) MNIST ($k = 10$) by binary SVMs

(l) MNIST ($k = 10$) by binary random forests

(m) letter ($k = 26$) by binary SVMs

(n) letter ($k = 26$) by binary random forests

Figure 8: Log likelihood (37) by using four probability estimates methods based on binary SVMs (left) and binary random forests (right). MSE of δ_V is too small and is not presented. solid line: 300 training/500 testing points; dotted line: 800 training/1,000 testing points.

G. W. Brier. Verification of forecasts expressed in probabilities. *Monthly Weather Review*, 78:1–3, 1950.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

K. Duan and S. S. Keerthi. Which is the best multiclass SVM method? An empirical study. Technical Report CD-03-12, Control Division, Department of Mechanical Engineering, National University of Singapore, 2003.

J. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996. Available at `http://www-stat.stanford.edu/reports/friedman/poly.ps.Z`.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1):451–471, 1998.

J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

D. R. Hunter. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32:386–408, 2004.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. MNIST database available at `http://yann.lecun.com/exdb/mnist/`.

A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2/3:18–22, December 2002. URL `http://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf`.

H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on Platt's probabilistic outputs for support vector machines. Technical report, Department of Computer Science, National Taiwan University, 2003. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/plattprob.ps`.

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`.

J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000. MIT Press. URL `citeseer.nj.nec.com/platt99probabilistic.html`.

D. Price, S. Knerr, L. Personnaz, and G. Dreyfus. Pairwise nerual network classifiers with probabilistic outputs. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Neural Information Processing Systems*, volume 7, pages 1109–1116. The MIT Press, 1995.

P. Refregier and F. Vallet. Probabilistic approach for multiclass classification with neural networks. In *Proceedings of International Conference on Artificial Networks*, pages 1003–1007, 1991.

S. Ross. *Stochastic Processes*. John Wiley & Sons, Inc., second edition, 1996.

V. Sventnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. Random forest: a tool for classification and regression in compound classification and QSAR modeling. *Journal of Chemical Information and Computer Science*, 43(6):1947–1958, 2003.

T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/svmprob.pdf`.

E. Zermelo. Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29:436–460, 1929.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–134, 2004.

Journal of Machine Learning Research 5 (2004) 1007–1034 Submitted 6/03; Revised 05/04; Published 8/04

On Robustness Properties of Convex Risk Minimization Methods for Pattern Recognition

Andreas Christmann CHRISTMANN@STATISTIK.UNI-DORTMUND.DE
University of Dortmund
Department of Statistics
44221 Dortmund, Germany

Ingo Steinwart INGO@LANL.GOV
Modeling, Algorithms and Informatics Group, CCS-3
Mail Stop B256
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

Editor: Peter Bartlett

Abstract

The paper brings together methods from two disciplines: machine learning theory and robust statistics. We argue that robustness is an important aspect and we show that many existing machine learning methods based on the convex risk minimization principle have − besides other good properties − also the advantage of being robust. Robustness properties of machine learning methods based on convex risk minimization are investigated for the problem of pattern recognition. Assumptions are given for the existence of the influence function of the classifiers and for bounds on the influence function. Kernel logistic regression, support vector machines, least squares and the AdaBoost loss function are treated as special cases. Some results on the robustness of such methods are also obtained for the sensitivity curve and the maxbias, which are two other robustness criteria. A sensitivity analysis of the support vector machine is given.

Keywords: AdaBoost loss function, influence function, kernel logistic regression, robustness, sensitivity curve, statistical learning, support vector machine, total variation

1. Introduction

In statistical learning theory the principle of regularized empirical risk minimization based on convex loss functions plays an important role, see Vapnik (1998). One strong argument in favor of such methods is that many classifiers based on convex loss functions are universally consistent under weak conditions. Nevertheless, it is important to investigate robustness properties for such statistical learning methods for the following reasons. In almost all cases statistical models are only approximations to the true random process which generated a given data set and for which a method for analyzing the data is designed. Hence the natural question arises what impact such deviations may have on the results. J.W. Tukey, one of the pioneers of robust statistics, mentioned already in 1960 (Hampel et al., 1986, p. 21):

> *A tacit hope in ignoring deviations from ideal models was that they would not matter; that statistical procedures which were optimal under the strict model would still be approximately optimal under the approximate model. Unfortunately, it turned out that*

> *this hope was often drastically wrong; even mild deviations often have much larger effects than were anticipated by most statisticians.*

The main aims of robust statistics are the description of the structure best fitting the *bulk* of the data and the identification of points deviating from this structure or deviating substructures for further treatment, *cf.* Hampel et al. (1986). Very briefly, a good robust method can be described as follows.

- If the strict model assumptions are violated, than the results of a robust method are only influenced in a moderate way by a few data points, which deviate grossly from the structure of the bulk of the data set, or by many data points, which deviate only mildly from the structure of the bulk of the data set.
- A robust method should have a reasonable high efficiency when the data set was in fact generated by the assumed model.

In practice one has to apply machine learning methods to a data set with a finite sample size. Machine learning methods are nonparametric tools. Nevertheless, the robustness issue is important, because the classical assumption that all data points were *independently* generated by the *same* distribution can be violated in practice. One reason is that outliers often occur in real data sets. Outliers can be described as data points which "are far away ... from the pattern set by the majority of the data", see Hampel et al. (1986, p. 25). Sometimes outliers are even correlated, which contradicts the classical assumption that the observations in the data set were generated in an independent manner. There are many reasons for the occurrence of outliers, *e.g.* typing errors and gross errors, which are errors due to a source of deviations which acts only occasionally but is quite powerful. *E.g.* undetected outliers may have an extreme impact on the estimation of insurance tariffs computed by motor vehicle insurance companies. From a robustness point of view the occurrence of outliers is only one of several possible deviations from the assumed model. There are often no or virtually no gross errors in high-quality data, but 1% to 10% of gross errors in routine data seem to be more the rule than the exception, *cf.* Hampel et al. (1986, p. 27f). Especially in large data mining problems the data quality is sometimes far from being optimal, *cf.* Hand et al. (2001) or Hipp et al. (2001). Obviously, it is *not* the goal to *model* the occurrence of typing errors or gross errors, because it is unlikely that they will occur in the same manner for other data sets which will be collected in the future. Goals of robust statistics are to investigate the impact such data points can have on the results of estimation, testing or prediction methods and the development of methods such that the impact of such data points is bounded.

We like to give an example showing that robustness properties of statistical methods can be very important in practice. A data set from 15 German motor vehicle insurance companies from the Verband öffentlicher Versicherer in Düsseldorf, Germany, is investigated by Christmann (2004). The main goals are the estimation of the expected claim amount, which is the primary response variable, and the probability that a customer has at least one claim within one year, which is the secondary response variable. It is well-known that even the very weak assumptions typically made by machine learning methods, see Section 2, may be violated for such data sets for the following reasons. The true values of the claim amount, *i.e.* the primary response variable, is not known exactly for all customers. *E.g.* if a major accident occurs in November, the exact claim size will often not be known at the end of the year and perhaps not even at the end of the following year. Possible reasons are law-suits or the case of physical injuries. In this case, a statistician will have to use more or less appropriate estimations of the exact claim size to construct a new insurance tariff

for the next year. Hence, the empirical distribution of the claim amount is in general a mixture of really observed values and of estimated claim amounts. Further, some explanatory variables may have imprecise values. *E.g.* there is a variable describing how many kilometers a customer is driving with the car within one year. The customer has to choose between some categories, *e.g.* below 9000 kilometers, between 9000 and 12000 kilometers, between 12000 and 15000 kilometers, and so on. There are reasons making it plausible, that a percentage of these values in the data set are too small, *e.g.* it is well-known to the customers that the premium of an insurance tariff increases for increasing values of this variable. The data set contains data from more than four million customers with more than 70 variables. Hence there is a high probability that some data points are typing errors, although the data set is of high quality.

There are different approaches to robustness in the statistical literature. For statistics representable as a functional of the empirical distribution, qualitative robustness, which is defined as equicontinuity of the distributions of the statistic as the sample size changes, is closely related to continuity of the statistic viewed as a functional in the weak(-star) topology, *cf.* Huber (1981, p. 7f) or Hampel et al. (1986, p. 41). The concept of qualitative robustness is a rather weak robustness criterion. It has the disadvantage that it does not offer arguments how to choose among different qualitative robust procedures. Huber's minimax approach of robust statistics (Huber, 1964, 1981) is to minimize the maximum asymptotic variance of the estimator within a neighborhood of the model. Other strategies of robust statistics are Hampel's influence function (Hampel, 1974; Hampel et al., 1986), the finite sample breakdown point proposed by Donoho and Huber (1983), the approach based on least favourable local alternatives (Rieder, 1994), and the regression depth method proposed by Rousseeuw and Hubert (1999).

Here, we will mainly use the approach based on the influence function. This approach can be applied to quite general models and the influence function has a nice interpretation, because it is a special Gâteaux derivative, see Section 3. A map T is called robust in the theory of robust statistics based on influence functions, if T has as a *bounded* influence function. From the viewpoint of robust statistics it is important to investigate the impact a small amount of contamination of the 'true' probability measure $\mathbb{P}$ can have on the statistical learning process which is specified via the regularized theoretical risk, *i.e.* the objective functions $R^{reg}_{L,\mathbb{P},\lambda}(.)$ and $R^{reg}_{L,\mathbb{P},\lambda}(.,.)$ given in (6) and (7).

This paper investigates robustness properties of statistical learning methods based on convex risk minimization and is organized as follows. Section 2 gives some notions on convex risk minimization methods. Section 3 gives the definitions of the influence function, the sensitivity curve, and the maxbias, which are the robustness concepts we are dealing with. Section 4 and Section 5 contain the main results. For practical applications Theorem 12 is our most important result and it covers a broad class of loss functions including the ones used by SVM, kernel logistic regression, AdaBoost and least squares. In Section 4 sufficient conditions are given for the existence of the influence function for classifiers based on (6) and (7). In Section 5 it is shown that the influence function of the solution of (7) and the difference quotient used in the definition of the influence function for (6) can be bounded independently of z and $\mathbb{P}$. Bounds for the sensitivity curve and for the maxbias are also given. Section 6 describes the results of some simulation experiments to gain insight into the robustness properties of the SVM for finite sample sizes and investigates the impact a single data point can have if a radial basis function kernel or a linear kernel is used. Section 7 contains the conclusion. Finally, the Appendix gives the proofs of the main theorems discussed in this paper and lists some mathematical facts which are used in our proofs.

2. Convex Risk Minimization in Machine Learning

In pattern recognition and statistical machine learning the major goal is the estimation of a functional relationship $y_i \approx f(x_i)$ between an outcome y_i and a vector of explanatory variables $x_i = (x_{i,1},\ldots,x_{i,k})' \in \mathbb{R}^d$. The function f is unknown. The estimate for f is used to get predictions of an unobserved outcome y_{new} based on an observed value x_{new}. One needs the implicit assumption that the relationship between x_{new} and y_{new} is—at least almost—the same as in the training data set (x_i,y_i), $i = 1,\ldots,n$. Otherwise, it is useless to extract knowledge on f from the training data set. The classical assumption in machine learning is that the training data (x_i,y_i) are independent and identically generated from an underlying unknown distribution $\mathbb{P}$ for a pair of random variables (X_i,Y_i). In practical applications the training data set is often quite large, high dimensional and complex. The quality of the predictor $f(x_i)$ is measured by some loss function $L(y_i, f(x_i))$. The goal is to find a predictor $f_{\mathbb{P}}(x_i)$ that minimizes the expected loss, *i.e.*

$$\mathbb{E}_{\mathbb{P}} L(Y, f_{\mathbb{P}}(X)) = \min_f \mathbb{E}_{\mathbb{P}} L(Y, f(X)), \tag{1}$$

where $\mathbb{E}_{\mathbb{P}} L(Y, f(X)) = \int L(y, f(x)) d\mathbb{P}(x,y)$ denotes the expectation of L with respect to $\mathbb{P}$. We sometimes write $L(f)$ instead of $L(y, f(x))$ and $L(f+b)$ instead of $L(y, f(x)+b)$ to shorten the notation, if misunderstandings are unlikely. We use this kind of notation also for derivatives of L.

In this paper we are interested in binary classification, where $y_i \in Y := \{-1,+1\}$. The straightforward prediction rule is: predict $y_i = +1$ if $f(x_i) \geq 0$, and predict $y_i = -1$ otherwise. The loss function for the classification error is given by $I(y_i, f(x_i)) = \mathbb{I}(y_i f(x_i) < 0) + \mathbb{I}(f(x_i) = 0)\mathbb{I}(y_i = -1)$, where $\mathbb{I}$ denotes the indicator function. Inspired by the law of large numbers one might estimate $f_{\mathbb{P}}$ with the minimizer f_{emp} of the empirical classification error, that is

$$f_{emp} = \arg\min_f \frac{1}{n} \sum_{i=1}^n I(y_i, f(x_i)) \ . \tag{2}$$

To avoid over-fitting one usually has to restrict the class of functions f considered in (2). Unfortunately, the classification function I is not convex and the minimization of (2) is often NP-hard, *cf.* Höffgen et al. (1995). To circumvent this problem, one can replace the classification error function $I(y_i, f(x_i))$ in (2) by a convex upper bound $L : Y \times \mathbb{R} \to \mathbb{R}$ *cf.* Vapnik (1998) and Schölkopf and Smola (2002). Furthermore, using reproducing kernel Hilbert spaces and an additional regularization term have some algorithmic advantages. These modifications lead to the following empirical regularized risks:

$$\hat{f}_{n,\lambda} = \arg\min_{f \in H} \lambda \| f \|_H^2 + \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)), \tag{3}$$

$$(\hat{f}_{n,\lambda}, \hat{b}_{n,\lambda}) = \arg \min_{f \in H, b \in \mathbb{R}} \lambda \| f \|_H^2 + \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i)+b), \tag{4}$$

where $\lambda > 0$ is a small regularization parameter, H is a reproducing kernel Hilbert space (RKHS) of a kernel k, and b is called *offset*. The decision functions are $\text{sign}(\hat{f}_{n,\lambda})$ or $\text{sign}(\hat{f}_{n,\lambda} + \hat{b}_{n,\lambda})$. Note that in practice usually (4) is solved while many theoretical papers deal with (3) since the unregularized offset b often causes technical difficulties in the analysis.

In practice the dual problems of (3) and (4) are solved. In these problems the RKHS does not occur explicitly, instead the corresponding kernel is involved. The choice of the kernel k enables

Method	L	L'
Support Vector Machine	$\max(1-v,0)$	-1, if $v<1$ 0, if $v>1$
Kernel Logistic Regression	$\ln(1+\exp(-v))$	$-1/(1+\exp(v))$
AdaBoost	$\exp(-v)$	$-\exp(-v)$
Least Squares	$(1-v)^2$	$2(v-1)$
Modified Least Squares	$\max(1-v,0)^2$	$-2\max(0,1-v)$
Modified Huber	$-4v$, if $v<-1$ $\max(1-v,0)^2$, else	-4, if $v<-1$ $-2\max(0,1-v)$, otherwise

Table 1: Loss functions, where $v=yf(x)$ or $v=y[f(x)+b]$, respectively.

the above methods to efficiently estimate not only linear, but also non-linear functions. Of special importance is the Gaussian radial basis function (RBF) kernel

$$k(x,x') = \exp(-\gamma\|x-x'\|^2), \quad \gamma>0, \tag{5}$$

which is a universal kernel on every compact subset of $\mathbb{R}^d$ in the sense of Steinwart (2001). Furthermore, this kernel is a bounded kernel, as $|k(x,x')|\le 1$ for all $x,x'\in\mathbb{R}^d$. Polynomial kernels $k(x,x')=(c+\langle x,x'\rangle)^m$ are also popular in practice, but are unbounded for $m\ge 1$ and $X=\mathbb{R}^d$.

Popular loss functions depend on y and f via $v=yf(x)$ or $v=y(f(x)+b)$. Some important specifications of L are given in Table 1. The support vector machine (SVM) penalizes points linearly if $v<1$. Kernel logistic regression and AdaBoost use twice continuously differentiable loss functions. The loss function used by kernel logistic regression (Wahba, 1999) penalizes misclassifications in a similar way to the SVM, *i.e.* approximately linearly if $v\to-\infty$. The loss function used by AdaBoost increases exponentially for $v\to-\infty$, *cf.* Freund and Schapire (1996), Friedman et al. (2000), and Hastie et al. (2001). The modified Huber's loss function, *cf.* Zhang (2004), changes the modified least squares loss such that misclassified points with $v<-1$ are penalized only linearly.

Two major benefits of using a convex loss function are known:

- For convex loss functions the resulting problems (3) and (4) are *computationally tractable* in the sense that they can be approximately solved in polynomial time. In fact, for many loss functions fast algorithms do exist. For bounded loss functions the convexity is lost and to our best knowledge almost nothing is known whether the problems are computational tractable. For applications of non-convex loss functions in the context of weighted least squares support vector machines for regression problems see Suykens et al. (2002).

- In the last two years an exciting observation almost revolutionized considerations on the learning performance of classification algorithms. The standard approach for bounding the estimation error of (regularized) empirical risk minimization (ERM) algorithms is to apply a uniform deviation bound. With this technique no learning rates faster than $n^{-\frac{1}{2}}$, where n is the sample size, can be obtained for nontrivial function classes and noisy distributions. However, if one "quantifies" the amount of noise (Tsybakov, 2004) and considers ERM-type algorithms with *convex* loss functions then learning rates up to n^{-1} are possible! For more information of this recent development we refer to Bartlett et al. (2002), Bartlett et al. (2003) and Bartlett and Mendelson (2002). In particular, this program has been successively applied to support

vector machines by Scovel and Steinwart (2003). Learning rates for boosting methods are investigated by Blanchard et al. (2003).

These two benefits of convex loss functions are the major reasons why the convexity plays a very important role in recent machine learning algorithms. This is in contrast to robust statistics, where often non-convex loss functions are used, although such robust statistics are based on objective functions with more than one local optimum, *c.f.* Hampel et al. (1986) and Christmann (1994, 1998).

Problems (3) and (4) can be interpreted as a stochastic approximation of the minimization of the theoretical regularized risk given in (6) or (7), respectively (Vapnik, 1998; Zhang, 2004; Steinwart, 2002a):

$$f_{\mathbb{P},\lambda} = \arg\min_{f \in H} \lambda \| f \|_H^2 + \mathbb{E}_{\mathbb{P}} L(Y, f(X)), \tag{6}$$

$$(f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda}) = \arg \min_{f \in H, b \in \mathbb{R}} \lambda \| f \|_H^2 + \mathbb{E}_{\mathbb{P}} L(Y, f(X) + b). \tag{7}$$

The objective functions in (6) and (7) are denoted by $R^{reg}_{L,\mathbb{P},\lambda}(.)$ and $R^{reg}_{L,\mathbb{P},\lambda}(.,.)$ in the sequel.

Steinwart (2002b) shows that SVM's are universally consistent, *i.e.* the classification error of $\hat{f}_{n,\lambda}(.)$ converges to the optimal Bayes error $\mathbb{E}_{\mathbb{P}} I(Y, f_{\mathbb{P}}(X))$ in probability, provided that the reproducing kernel Hilbert space is dense in the space $C(X)$, $X \subset \mathbb{R}^d$ compact, and $\lambda = \lambda_n$ tends "slowly" to 0 for $n \to \infty$. Zhang (2004) improves this result by showing that for many convex loss functions the classifiers based on (3) are universally consistent if $\lambda_n \to 0$ and $\lambda_n n \to \infty$. Steinwart (2002a) characterizes the loss functions which lead to universally consistent classifiers and establishes universal consistency for classifiers based on (3) and (4). Furthermore, he shows that there exist solutions of both the theoretical and the empirical problems. Under certain assumptions on the data generating distribution one can even establish rates on the learning speed of SVM's. Such results can be found in Steinwart (2001), Chen et al. (2003), and Scovel and Steinwart (2003). Moreover, Steinwart (2003) gives lower asymptotical bounds on the number of support vectors, *i.e.* on the data points with non-vanishing coefficients, and investigates the asymptotic behavior of $\hat{f}_{n,\lambda}(.)$ in terms of the loss function L. As a by-product it also turns out that the solutions of (3) and (6) are unique. The same holds for the RKHS part of the solutions of (4) and (7). Upper bounds on the number of support vectors can be found in Steinwart (2004). Schölkopf and Smola (2002) describe other support vector machines and give an overview on algorithms to solve the minimization problems corresponding to SVMs.

3. Robustness

In the statistical literature different criteria have been proposed to define the notion of robustness in a mathematical way, *e.g.* the minimax approach (Huber, 1964), the sensitivity curve (Tukey, 1977), the approach based on influence functions (Hampel, 1974; Hampel et al., 1986), the maxbias curve (Huber, 1964; Hampel et al., 1986), and the finite sample breakdown point (Donoho and Huber, 1983).

In this paper, we mainly use the approach based on the influence function proposed by Hampel (1974) and Hampel et al. (1986). We will consider a map T which assigns to every distribution $\mathbb{P}$ on a given set Z an element $T(\mathbb{P})$ of a given Banach space E. In the case of the convex risk minimization methods given in (6) and (7) E equals the RKHS and $T(\mathbb{P}) = f_{\mathbb{P},\lambda}$ or $T(\mathbb{P}) = (f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$,

respectively. The book by Huber (1981, p. 34ff) is a standard reference for Gâteaux and Fréchet derivatives in the context of robust statistics.

Definition 1 (Influence function) *The influence function of T at a point z for a distribution $\mathbb{P}$ is the special Gâteaux derivative (if it exists)*

$$IF(z;T,\mathbb{P}) = \lim_{\varepsilon \downarrow 0} \frac{T\big((1-\varepsilon)\mathbb{P} + \varepsilon\Delta_z\big) - T(\mathbb{P})}{\varepsilon} \quad , \tag{8}$$

where Δ_z is the Dirac distribution at the point z such that $\Delta_z(\{z\}) = 1$.

The influence function has the interpretation, that it measures the impact of an (infinitesimal) small amount of contamination of the original distribution $\mathbb{P}$ in direction of a Dirac distribution located in the point z on the theoretical quantity of interest $T(\mathbb{P})$. Therefore, in the robustness approach based on influence functions it is desirable that a statistical method which can be written as $T(\mathbb{P})$ has a *bounded* influence function. If T fulfills some regularity conditions, it can be linearized near $\mathbb{P}$ in terms of the influence function via

$$T(\mathbb{P}^*) = T(\mathbb{P}) + \int IF(z;T,\mathbb{P})\,[\mathbb{P}^*(dz) - \mathbb{P}(dz)] + \dots ,$$

where $\mathbb{P}^*$ is a probability distribution near $\mathbb{P}$, *cf.* Huber (1981, p. 14). If different methods have a bounded influence function, the one with a lower bound is the more robust one.

The sensitivity curve SC_n proposed by J.W. Tukey and discussed in detail by Hampel et al. (1986, p. 93) can be interpreted as a finite sample version of the influence function, see (10). The sensitivity curve measures the impact of just one additional data point z on the empirical quantity of interest, *i.e.* on the estimate T_n.

Definition 2 (Sensitivity curve) *The sensitivity curve of an estimator T_n at a point z given a data set $z_1,\dots,z_{n-1}$ is defined by*

$$SC_n(z;T_n) = n\big(T_n(z_1,\dots,z_{n-1},z) - T_{n-1}(z_1,\dots,z_{n-1})\big)\,. \tag{9}$$

If the estimator T_n is defined via $T(\mathbb{P}_n)$, where $\mathbb{P}_n$ denotes the empirical distribution of the data points $z_1,\dots,z_n$, then we have for $\varepsilon_n = 1/n$:

$$SC_n(z;T_n) = \frac{T\big((1-\varepsilon_n)\mathbb{P}_{n-1} + \varepsilon_n\Delta_z\big) - T(\mathbb{P}_{n-1})}{\varepsilon_n}\,. \tag{10}$$

Of theoretical as well as of practical importance is also the notion of maxbias, which measures the maximum bias $T(Q) - T(\mathbb{P})$ within a neighborhood of probability distributions Q near $\mathbb{P}$. In robust statistics the so-called contamination (or gross-error) neighborhood (defined below), *cf.* Huber (1981), is quite common for the following three reasons. It allows a good interpretation because it contains mixture distributions with respect to the 'true' distribution $\mathbb{P}$ and some other distribution. The contamination neighborhood has some relationship to the influence function and to breakdown points, and finally it is often easier to deal with this set of distributions than with other neighborhoods. Note that the contamination neighborhood is not a neighborhood in the topological sense.

Definition 3 (Maxbias) *Let $\mathbb{P}$ be a fixed probability distribution on $X \times Y$. A contamination neighborhood of $\mathbb{P}$ is given by*

$$N_\varepsilon(\mathbb{P}) = \left\{ Q = (1-\varepsilon)\mathbb{P} + \varepsilon\tilde{\mathbb{P}};\ \tilde{\mathbb{P}} \text{ is any distribution on } X \times Y,\ 0 \le \varepsilon < \frac{1}{2} \right\}.$$

The maxbias (or supremum bias) of T at the distribution $\mathbb{P}$ with respect to the contamination neighborhood $N_\varepsilon(\mathbb{P})$ is defined by

$$\operatorname{maxbias}(\varepsilon; T, \mathbb{P}) = \sup_{Q \in N_\varepsilon(\mathbb{P})} \|T(Q) - T(\mathbb{P})\|.$$

4. Existence of the Influence Function

In this section we give sufficient conditions for the existence of the influence function for classifiers based on (6) and (7), whereas the next section will show that the influence function is bounded under weak conditions. Most of our results are valid for *any* distribution $\mathbb{P}$ on $X \times Y$. Therefore, they are also valid for the special case of the empirical distribution $\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^n \Delta_{(x_i,y_i)}$, *i.e.* for a given data set, and for the empirical regularized risks defined in (3) and (4).

Since our robustness results in this section are based on the calculus in (infinite dimensional) Banach spaces we first recall some basic notions; more details are given in the appendix. To this end let $G : E \to F$ be a map between two Banach spaces E and F. We say that G is *(Fréchet)-differentiable in $x_0 \in E$* if there exists a bounded linear operator $A : E \to F$ and a function $\varphi : E \to F$ with $\frac{\varphi(x)}{\|x\|} \to 0$ for $x \to 0$ such that

$$G(x_0 + x) - G(x_0) \;=\; Ax + \varphi(x) \tag{11}$$

for all $x \in E$. It turns out that A is uniquely determined by (11). We hence write $G'(x) := \frac{\partial G}{\partial E}(x) := A$. The map G is called continuously differentiable if the map $x \mapsto G'(x)$ exists on E and is continuous. Analogously we define continuous differentiability on open subsets of E.

We also have to introduce the notion of Bochner-integrals. For simplicity we restrict ourselves to the RKHS case, since this is the only one we actually need. To this end let H be a RKHS of a bounded, continuous kernel k on X with feature map $\Phi : X \to H$, i.e. $\Phi(x) = k(x,.)$. Furthermore, let $\mathbb{P}$ be a probability measure on $X \times Y$ and $h : Y \times X \to \mathbb{R}$ be a function which is continuous in its second variable $x \in X$. Then the Bochner-integral $\mathbb{E}_\mathbb{P} h(Y,X)\Phi(X)$ is an element of H which in our case can be computed by a simple Riemann approach, i.e. by partitioning the underlying space $X \times Y$. For a precise definition of Bochner-integrals we refer to Diestel and Uhl (1977). Note that in our special situation we can also interpret $\mathbb{E}_\mathbb{P} h(Y,X)\Phi(X)$ as an element of the dual space H' by the Fréchet-Riesz theorem, i.e. $\mathbb{E}_\mathbb{P} h(Y,X)\Phi(X)$ acts as a functional on H via $w \mapsto \langle \mathbb{E}_\mathbb{P} h(Y,X)\Phi(X), w\rangle$. Finally, we have to consider Bochner-integrals of the form $\mathbb{E}_\mathbb{P} h(Y,X)\langle \Phi(X), .\rangle\Phi(X)$ which define bounded linear operators on H by the map $w \mapsto \mathbb{E}_\mathbb{P} h(Y,X)\langle \Phi(X), w\rangle\Phi(X)$.

We can now establish our first two results which treat classifiers based on (6) with a smooth loss function and a bounded continuous kernel. The first theorem covers *e.g.* the Gaussian RBF kernel. The Dirac distribution in the point z is denoted by Δ_z.

Theorem 4 *Let $L : Y \times \mathbb{R} \to [0,\infty)$ be a convex and twice continuously differentiable loss function. Furthermore, let $X \subset \mathbb{R}^d$ be a closed or open subset, H be a RKHS of a bounded continuous kernel*

on X and $\mathbb{P}$ be a distribution on $X \times Y$. We define $G : \mathbb{R} \times H \to H$ by

$$G(\varepsilon, f) := 2\lambda f + \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z} L'(Y, f(X))\Phi(X)$$

which implies

$$\frac{\partial G}{\partial \varepsilon}(0, f_{\mathbb{P},\lambda}) = -\mathbb{E}_{\mathbb{P}}[L'(Y, f_{\mathbb{P},\lambda}(X))\Phi(X)] + L'(z_y, f_{\mathbb{P},\lambda}(z_x))\Phi(z_x).$$

Furthermore, we define $S : H \to H$ by

$$S := \frac{\partial G}{\partial H}(0, f_{\mathbb{P},\lambda}) = 2\lambda \mathrm{id}_H + \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X))\langle \Phi(X), . \rangle \Phi(X)\,.$$

Then the influence function of the classifiers based on (6) exists for all $z = (z_x, z_y) \in X \times Y$ and is given by

$$IF(z; T, \mathbb{P}) = -S^{-1} \circ \frac{\partial G}{\partial \varepsilon}(0, f_{\mathbb{P},\lambda})\,. \tag{12}$$

Remark 5 *The influence function derived in Theorem 4 depends on the point $z = (z_x, z_y)$, where the point mass contamination takes place, only by the term $L'(z_y, f_{\mathbb{P},\lambda}(z_x))\Phi(z_x)$.*

In practice the set X is usually a bounded and closed subset of $\mathbb{R}^d$ and hence compact. In this case existence of the influence function can be shown without the assumption that the kernel is bounded, and hence the following theorem covers also polynomial kernels.

Theorem 6 *Let $L : Y \times \mathbb{R} \to [0, \infty)$ be a convex and twice continuously differentiable loss function. Furthermore, let $X \subset \mathbb{R}^d$ be compact, H be a RKHS of a continuous kernel on X and $\mathbb{P}$ be a distribution on $X \times Y$. Then the influence function of the classifiers based on (6) exists for all $z \in X \times Y$.*

Remark 7 *By a simple modification of the proof of the above theorem we actually find that the special Gâteaux derivative of $T : \mathbb{P} \mapsto f_{\mathbb{P},\lambda}$ exists for every direction,* i.e.

$$\lim_{\varepsilon \downarrow 0} \frac{f_{(1-\varepsilon)\mathbb{P}+\varepsilon\tilde{\mathbb{P}},\lambda} - f_{\mathbb{P},\lambda}}{\varepsilon}$$

exists for all distributions $\mathbb{P}$ and $\tilde{\mathbb{P}}$ on $X \times Y$ provided that the assumptions of Theorem 4 hold. This is an interesting result from the viewpoint of applied statistics, because a point mass contamination is just one possible kind of contamination which can occur in practice.

The following theorem shows the existence of the influence function for classifiers based on (7). Since the offset $b_{\mathbb{P},\lambda}$ can be infinite for certain loss functions if $\mathbb{P}$ is degenerate, i.e $\mathbb{P}(\{(y,x) : x \in X\}) = 1$ for $y = +1$ or $y = -1$, we have to exclude these probability measures. Note that these measures almost surely produce training sets of the form $((1,x_1),\ldots,(1,x_n))$, or $((-1,x_1),\ldots,(-1,x_n))$, respectively.

Theorem 8 *Let $L : Y \times \mathbb{R} \to [0,\infty)$ be a convex and twice continuously differentiable loss function with $L'' > 0$. Furthermore, let $X \subset \mathbb{R}^d$ be open or closed, H be a RKHS of a continuous kernel on X and $\mathbb{P}$ be a non-degenerate distribution on $X \times Y$. We define $G : \mathbb{R} \times H \times \mathbb{R} \to H \times \mathbb{R}$ by*

$$G(\varepsilon, f, b) := \Big(2\lambda f + \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z} L'(Y, f(X)+b)\Phi(X), \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z} L'(Y, f(X)+b)\Big)$$

which implies

$$\frac{\partial G}{\partial \varepsilon}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda}) = -\mathbb{E}_{\mathbb{P}}[L'(Y, f_{\mathbb{P},\lambda}(X)+b_{\mathbb{P},\lambda})\Phi(X)] + L'(z_y, f_{\mathbb{P},\lambda}(z_x)+b_{\mathbb{P},\lambda})\Phi(z_x)\,.$$

Furthermore, for $S := \frac{\partial G}{\partial (H\times\mathbb{R})}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$ we have

$$S = \begin{pmatrix} 2\lambda \mathrm{id}_H + \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X)+b_{\mathbb{P},\lambda})\langle \Phi(X), .\rangle \Phi(X) & \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X)+b_{\mathbb{P},\lambda})\Phi(X) \\ \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X)+b_{\mathbb{P},\lambda})\Phi(X) & \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X)+b_{\mathbb{P},\lambda}) \end{pmatrix}.$$

Then the influence function of the classifiers based on (7) exists for all $z = (z_x, z_y) \in X \times Y$ and is given by

$$IF(z; T, \mathbb{P}) = -S^{-1} \circ \frac{\partial G}{\partial \varepsilon}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})\,. \tag{13}$$

Remark 9 *The influence function derived in Theorem 8 depends on the point $z = (z_x, z_y)$, where the point mass contamination takes place, only by the term $L'(z_y, f_{\mathbb{P},\lambda}(z_x)+b_{\mathbb{P},\lambda})\Phi(z_x)$. Hence loss functions L and kernels k such that L' and the feature map Φ are bounded are of special interest from the view point of robust statistics.*

Remark 10 *As in the case of problem (6) a slight modification of the proof gives that $T : \mathbb{P} \mapsto (f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$ is special Gâteaux differentiable.*

Remark 11 *Considering the loss functions in Table 1 we immediately see that the above theorems apply to the kernel logistic regression, the least squares and the AdaBoost loss function. The second derivatives of the modified least squares and the modified Huber loss function fail to exists in only one point. For the loss function of the standard SVM, even the first derivative does not exist in one point.*

5. Bounds on the Influence Function, Sensitivity Curve and Maxbias

As mentioned in Section 1, a desirable property of a robust statistical method is that T has a bounded influence function. In this section we show that for certain loss functions the influence function can be bounded *independently* of z and $\mathbb{P}$ for classifiers based on (6) and (7). For the formulation of our results we need to recall that the norm of total variation of a signed measure μ on a space X is defined by

$$\|\mu\|_{\mathcal{M}} := |\mu|(X) := \sup\Big\{\sum_{i=1}^{n} |\mu(A_i)| : A_1, \ldots, A_n \text{ is a partition of } X\Big\}.$$

For more information on this norm we refer to Brown and Pearcy (1977).

Our first result bounds the difference quotient in the definition of the influence function for classifiers based on (6). For practical applications Theorem 12 is our most important result. In particular,

it states that the influence function of these classifiers is uniformly bounded whenever it exists, and that the sensitivity curve is uniformly bounded too. Please note that the following theorem based on Steinwart (2003) applies to all six loss functions given in Table 1 because differentiability of L is not assumed.

Theorem 12 *Let $L : Y \times \mathbb{R} \to [0,\infty)$ be a continuous and convex loss function. Furthermore, let $X \subset \mathbb{R}^d$ and H be a RKHS of a bounded, continuous kernel on X. Then for all $\lambda > 0$ there exists a constant $c_L(\lambda) > 0$ explicitly given in (27) such that for all distributions $\mathbb{P}$ and $\tilde{\mathbb{P}}$ on $X \times Y$ we have*

$$\left\| \frac{f_{(1-\varepsilon)\mathbb{P}+\varepsilon\tilde{\mathbb{P}},\lambda} - f_{\mathbb{P},\lambda}}{\varepsilon} \right\|_H \leq c_L(\lambda)\, \|\mathbb{P} - \tilde{\mathbb{P}}\|_{\mathcal{M}}, \quad \varepsilon > 0\,.$$

Remark 13 *The above theorem also gives uniform bounds for Tukey's sensitivity curve of f. Consider the special case that $\mathbb{P}$ is equal to the empirical probability measure of $(n-1)$ data points,* i.e. *$\mathbb{P}_{n-1} = \frac{1}{n-1}\sum_{i=1}^{n-1} \Delta_{(x_i,y_i)}$, and that $\tilde{\mathbb{P}}$ is equal to the Dirac measure $\Delta_{(x,y)}$ in some point $(x,y) \in X \times Y$. Let $\varepsilon = \frac{1}{n}$. Under the assumptions of Theorem 12 it follows from (10), that*

$$n\| f_{(1-\varepsilon)\mathbb{P}_{n-1}+\varepsilon\Delta_{(x,y)},\lambda} - f_{\mathbb{P}_{n-1},\lambda}\|_H \leq c_L(\lambda)\, \|\mathbb{P}_{n-1} - \Delta_{(x,y)}\|_{\mathcal{M}}, \quad \varepsilon > 0\,.$$

Essentially, this result has already been established by Bousquet and Elisseeff (2002).

Remark 14 *Because no assumptions on $\tilde{\mathbb{P}}$ are made in Theorem 12, an upper bound for the maxbias of $f_{\mathbb{P},\lambda}$ (see Definition 3) for such machine learning methods is given by*

$$\operatorname{maxbias}(\varepsilon; f, \mathbb{P}) = \sup_{Q \in N_\varepsilon} \| f_{Q,\lambda} - f_{\mathbb{P},\lambda}\| \leq \varepsilon\, c_L(\lambda) \sup_{\tilde{\mathbb{P}} \in \mathcal{P}} \|\mathbb{P} - \tilde{\mathbb{P}}\|_{\mathcal{M}} \leq 2c_L(\lambda)\varepsilon,$$

where $\varepsilon \in (0, 1/2)$, and $\mathcal{P}$ denotes the set of all probability measures on $X \times Y$. As no assumptions are made for $\mathbb{P}$, this result is valid for empirical distributions too. Consider the empirical distribution $\mathbb{P}_n$ defined by given data set $(x_i, y_i) \in X \times Y$ with n data points. Then the maxbias of $f_{\mathbb{P}_n,\lambda}$ in a contamination neighborhood $N_\varepsilon(\mathbb{P}_n)$ is at most $2c_L(\lambda)\varepsilon$, where $\varepsilon \in (0, 1/2)$.

Unfortunately, using the estimate of Steinwart (2003) does not give any meaningful result for classifiers based on (7). However, under some additional assumptions on L we can still bound the influence function.

Theorem 15 *Let $L : Y \times \mathbb{R} \to [0,\infty)$ be a convex and twice continuously differentiable loss function with $a \leq L'' \leq b$ for some $a, b > 0$. Furthermore, let $X \subset \mathbb{R}^d$ be open or closed, H be a RKHS of a continuous kernel on X, and $T_\lambda(\mathbb{P}) = (f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$ be given by (7). Then for all $\lambda > 0$ there exists a constant $c_L(\lambda) > 0$ such that for all non-degenerated distributions $\mathbb{P}$ on $X \times Y$ and all $z \in X \times Y$ we have*

$$\|IF(z; T, \mathbb{P})\|_{H \times \mathbb{R}} \leq c_L(\lambda)\, \|\mathbb{P} - \Delta_z\|_{\mathcal{M}}\,.$$

Remark 16 *Theorem 15 applies to (7) with the least squares loss function. However, Theorem 15 covers neither the logistic regression loss function as we only have $L'' \geq 0$ nor the AdaBoost loss function which satisfies $L'' = L = \exp(-.)$ However, we get the same bound of the influence function if we restrict our considerations to distributions $\mathbb{P}$ with*

$$a \;\leq\; \int L''(Y, f_{\mathbb{P},\lambda}(X) + b_{\mathbb{P},\lambda}) d\mathbb{P} \;\leq\; b \tag{14}$$

for some $b \geq a > 0$. A simple sufficient condition for the latter can be derived by the proof of Steinwart (2002a, Lemma II.6): let $A_y^\rho := \{x \in X : \mathbb{P}(y|x) > \rho\}$, $y \in Y$, $\rho > 0$, and $\alpha_{\mathbb{P}}(\rho) := \rho \min\{\mathbb{P}_X(A_1^\rho), \mathbb{P}_X(A_{-1}^\rho)\}$. Then fixing $\lambda > 0$, a twice continuously differentiable L and a threshold $\alpha > 0$ there exist $b \geq a > 0$ such that every $\mathbb{P}$ with $\alpha_{\mathbb{P}}(\rho) \geq \alpha$ for some $\rho > 0$ satisfies (14). Note that the assumption $\alpha_{\mathbb{P}}(\rho) \geq \alpha$ guarantees that the two classes of $\mathbb{P}$ are "balanced".

Remark 17 *As mentioned in Remark 10 the map $T : \mathbb{P} \mapsto (f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$ is special Gâteaux differentiable. A simple modification of the proof of Theorem 15 shows that the special Gâteaux derivative of T can be uniformly bounded.*

Remark 18 *Consider the case that $\mathbb{P}$ and $\tilde{\mathbb{P}}$ are probability measures with densities p and $\tilde{p}$ with respect to some dominating measure ν. Then, the last two theorems also give bounds of the influence functions and the sensitivity curve in terms of the Hellinger metric $H(\mathbb{P}, \tilde{\mathbb{P}}) = [\int (\sqrt{p} - \sqrt{\tilde{p}})^2 d\nu]^{1/2}$. This follows from a relationship between the norm of total variation and the Hellinger metric:*

$$\|\mathbb{P} - \tilde{\mathbb{P}}\|_{\mathcal{M}} \leq 2H(\mathbb{P}, \tilde{\mathbb{P}}) \leq 2\|\mathbb{P} - \tilde{\mathbb{P}}\|_{\mathcal{M}}^{1/2}.$$

Note that the bounds for the difference quotient in Theorem 12 and for the influence function in Theorem 15 converge to infinity, if λ converges to 0 and $\|\mathbb{P} - \tilde{\mathbb{P}}\|_{\mathcal{M}} > 0$ or $\|\mathbb{P} - \Delta_z\|_{\mathcal{M}} > 0$. However, λ converging to 0 has the interpretation that misclassifications are penalized by constants C tending to ∞. Therefore, decreasing values of λ correspond to a decreasing amount of robustness, which was to be expected. The quantity λ can be interpreted as a tuning constant controlling the robustness properties of the method in a similar way than it is well-known for many robust methods, *e.g.* Huber-type M-estimators in location or regression models. Consider the Huber-type M-estimator (Huber, 1964) in a univariate location model, where all data points are realizations from n independent and identically distributed random variables with some distribution function $F(\cdot - \theta)$, where $\theta \in \mathbb{R}$ is unknown. Huber's robust M-estimator with tuning constant $b \in (0, \infty)$ has an influence function proportional to $\psi_b(z) = \min\{b, \max\{z, -b\}\}$, *cf.* Hampel et al. (1986, p. 104f). For all $b \in (0, \infty)$ the influence function is bounded by $\pm b$. However, the bounds tend to $\pm\infty$ if $b \to \infty$, and Huber's M-estimator with $b = \infty$ is equal to the non-robust maximum likelihood estimator which has an unbounded influence function. Therefore, the quantity $1/\lambda$ (or the cost C) in the machine learning methods we are dealing with has a similar role to the tuning constant b in Huber-type M-estimators.

6. Empirical Results for the SVM

In this section we study the impact an additional data point can have on the SVM with offset b for pattern recognition. An analogous investigation for the case without offset gave similar results to those described in this section. We generated a training data set with $n = 500$ data points x_i from a bivariate normal distribution with expectation $\mu = (0, 0)$ and covariance matrix Σ. The variances were set to 1, whereas the covariance was set to 0.5. The responses y_i were generated from a classical logistic regression model with $\theta = (-1, 1)'$, $b = 0.5$, such that $P(Y_i = +1) = [1 + \exp(-(x_i'\theta + b))]^{-1}$ and $P(Y_i = -1) = 1 - P(Y_i = +1)$. The computations were done using the software SVM^{light} developed by Joachims (1999). SVM^{light} solves the dual program corresponding to the primal optimization problem

$$\begin{array}{ll} \arg\min_{f \in H, b \in \mathbb{R}} & \frac{1}{2Cn}||f||_H^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i \\ \text{such that} & y_i(f(x_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0. \end{array} \tag{15}$$

We consider two popular kernels: a Gaussian radial basis function kernel with parameter γ, see (5) and a linear kernel. Appropriate values for γ and for the constant C (or λ) are important for the SVM and are often determined by cross validation, *cf.* Schölkopf and Smola (2002, p. 217). A cross validation based on the leave-one-out error for the training data set was carried out by a two-dimensional grid search on

$$\gamma \in \{0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 3, 4, 5, 10, 20\}$$

and

$$C \in \{0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 5, 10, 20\}.$$

As a result of the cross validation, the tuning parameters for the SVM with RBF kernel were set to $\gamma = 0.25$ and $C = 2$. The leave-one-out error for the SVM with a linear kernel turned out to be stable over a broad range of values for C. We used $C = 1$ in the computations for the linear kernel. For $n = 500$ this results in $\lambda = (2Cn)^{-1} = 5 \times 10^{-4}$ for the RFB kernel and $\lambda = (2Cn)^{-1} = 0.001$ for the linear kernel. Please note that such small values of λ will result in relatively large bounds.

Figure 1 shows the sensitivity curves of $\hat{f} + \hat{b} := \hat{f}_{n,\lambda} + \hat{b}_{n,\lambda}$, if we add a single point $z = (x, y)$ to the original data set, where $x_1 = 6$, $x_2 = 6$, and $y = +1$. The additional data point has a local and smooth impact on $\hat{f} + \hat{b}$ with a peak in a neighborhood of (x_1, x_2), if one uses the RBF kernel. For a linear kernel, the impact is approximately linear. The reason for this different behavior of the SVM with different kernels becomes clear from Figure 2 where plots of $\hat{f} + \hat{b}$ are given for the original data set and for the modified data set, which contains the additional data point z. Please note that the RBF kernel yields $\hat{f} + \hat{b}$ approximately equal to zero outside a central region, as almost all data points are lying inside the central region. Comparing the plots of $\hat{f} + \hat{b}$ based on the RBF kernel for the modified data set with the corresponding plot for the original data set, it is obvious that the additional smooth peak is due to the new data point located at $x = (6, 6)$ with $y = +1$. It is interesting to note that although the estimated functions $\hat{f} + \hat{b}$ for the original data set and for the modified data set based on the SVM with the linear kernel are looking quite similar, the sensitivity curve is similar to an affine hyperplane which is affected by the value of z. This allows the interpretation, that just a single data point can have an impact on $\hat{f} + \hat{b}$ estimated by a SVM with a linear kernel over a broader region than for an SVM with an RBF kernel.

Now, we study the impact of an additional data point $z = (x, y)$, where $y = +1$, on the percent of classification errors and on the fitted $y-$value for z. We vary z over a grid in the $x-$coordinates. Figure 3 shows that the percentage of classification errors is approximately constant outside the central region that contains almost all data points if a Gaussian RBF kernel was used. For the SVM with a linear kernel, the percentage of classification errors tends to be approximately constant in one half-space but changes in the other half-space. The response of the additional data point was correctly estimated by $\hat{y} = +1$ outside the central region, if a RBF kernel is used, see Figure 4. In contrast to that, using a linear kernel results in estimated responses $\hat{y} = +1$ or $\hat{y} = -1$ of the additional data point depending on the affine half-space in which the $x-$value of z is lying. Finally, let us study the impact of an additional data point located at $z = (x, y)$, where $y = +1$, on the estimated parameters $\hat{b}$ and $\hat{\theta}$, see Figure 5. We vary z over a grid in the $x-$coordinates in the same manner as before. As the plots for $\hat{\theta}_1$ and $\hat{\theta}_2$ are looking very similar, we only show the latter. Note that the axes are not identical in Figure 5 due to the kernels. The sensitivity curves for the slopes estimated by the SVM with an RBF kernel are similar to a hyperplane outside the central region, which contains almost all data points. In the central region, there is a smooth transition between

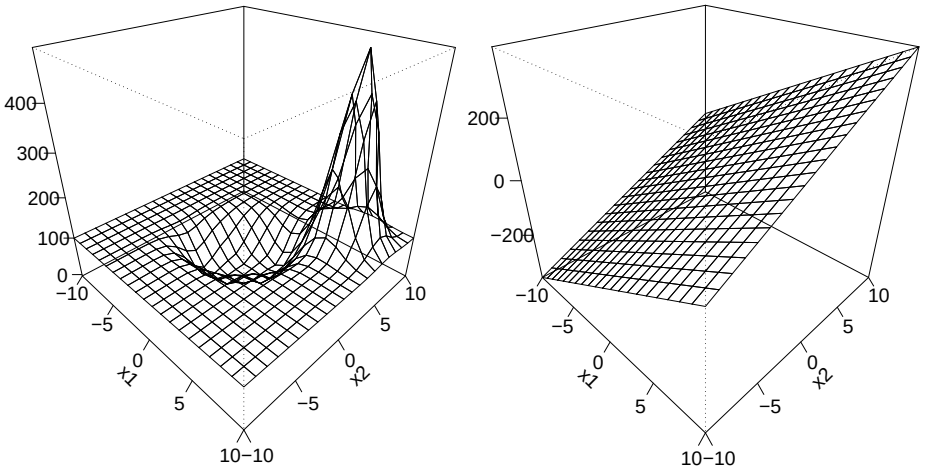


Figure 1: Sensitivity function of $\hat{f}+\hat{b}$, if the additional data point z is located at $z=(x,y)$, where $x=(6,6)$ and $y=+1$. Left: RBF kernel. Right: linear kernel.

regions with higher sensitivity values and regions with lower sensitivity values. The sensitivity curves for the slopes of the SVM with a linear kernel are flat in one affine half-space, but change approximately linearly in the other affine half-space. This behavior also occurs for the sensitivity curve of the offset by using a linear kernel. In contrast to that, the sensitivity curve of the offset based on a SVM with a RBF kernel shows a smooth but curved shape outside the region containing the majority of the data points.

7. Concluding Remarks

In this paper, we used the influence function approach of robust statistics (Hampel et al., 1986) for recent statistical learning methods based on convex risk minimization methods for the problem of pattern recognition. The influence function has the interpretation that it measures the impact of an infinitesimal amount of contamination of the original distribution $\mathbb{P}$ in direction of a Dirac distribution located in the point z on the theoretical quantity of interest $T(\mathbb{P})$. Special cases of such convex risk minimization methods are the support vector machine, kernel logistic regression, AdaBoost, and least squares. Assumptions were derived for the existence of the influence function of f or (f,b) used by the classifiers and also for uniform bounds on the influence function which hold with respect to the distribution $\mathbb{P}$ and the point z of the Dirac distribution Δ_z describing the contamination. For the case without offset b one can uniformly bound the difference quotient considered by the influence function under weak conditions which also yields uniform bounds for Tukey's sensitivity curve and uniform upper bounds for the maxbias. In particular, the influence function for these classifiers is uniformly bounded if it exists. Some of the results are not limited to the special Gâteaux derivative used in the definition of the influence function. The assumptions of some of our results exclude the support vector machine because the SVM uses a loss function which is not differentiable in one point, but Theorem 12 covers the SVM as a special case. We gave some numerical results for

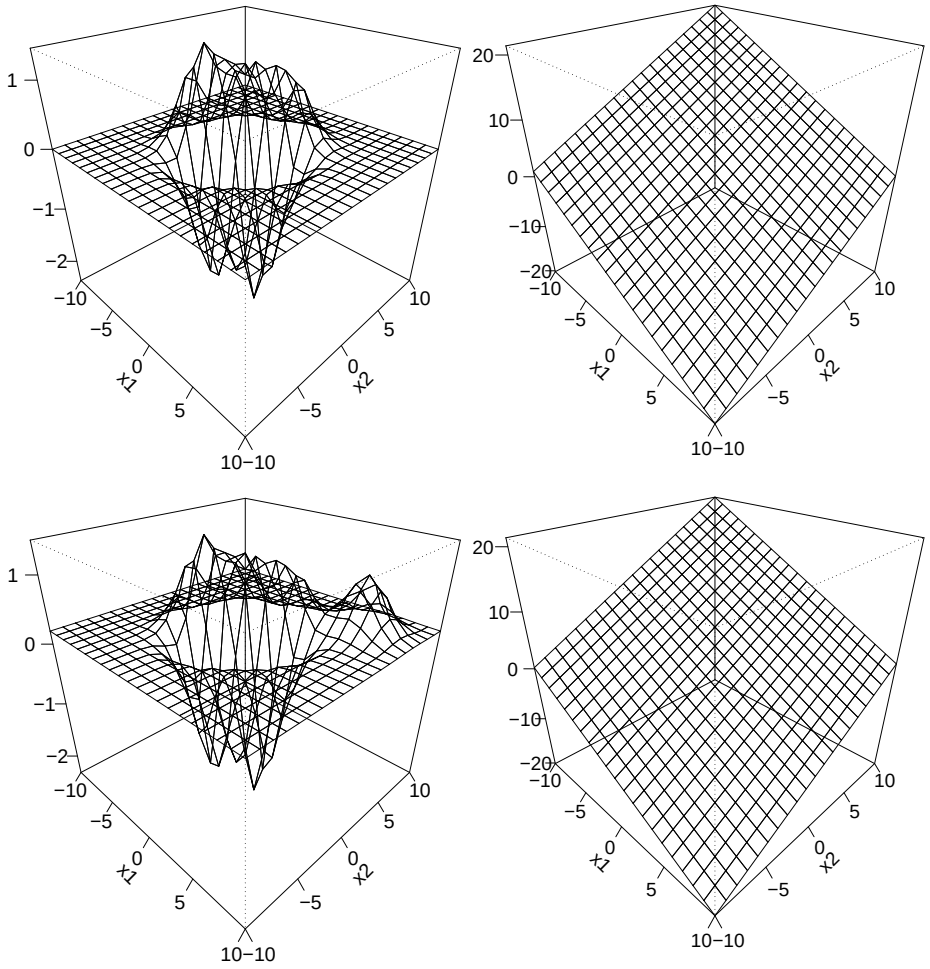


Figure 2: Plot of $\hat{f}+\hat{b}$. Upper left: RBF kernel, original data set. Upper right: linear kernel, original data set. Lower left: RFB kernel, modified data set. Lower right: linear kernel, modified data set. The modified data set contains the additional data point $z = (x, y)$, where $x = (6, 6)$ and $y = +1$.

the sensitivity curve, which can be interpreted as a finite sample version of the influence function, of the SVM classifier. It turned out, that the popular exponential radial basis function kernel resulted in smooth sensitivity curves for $\hat{f}+\hat{b}$ and for the estimated coefficients $(\hat{\theta}, \hat{b})$. Varying the position of one additional data point had a smooth and local impact on $\hat{f}+\hat{b}$, if one uses an RBF kernel. For the linear kernel the impact of varying one additional data point behaves also in a relatively smooth manner, but the impact seems to be more globally than locally.

For a numerical comparison between the support vector machine and the regression depth method recently proposed by Rousseeuw and Hubert (1999) see Christmann and Rousseeuw (2001) and Christmann et al. (2002).

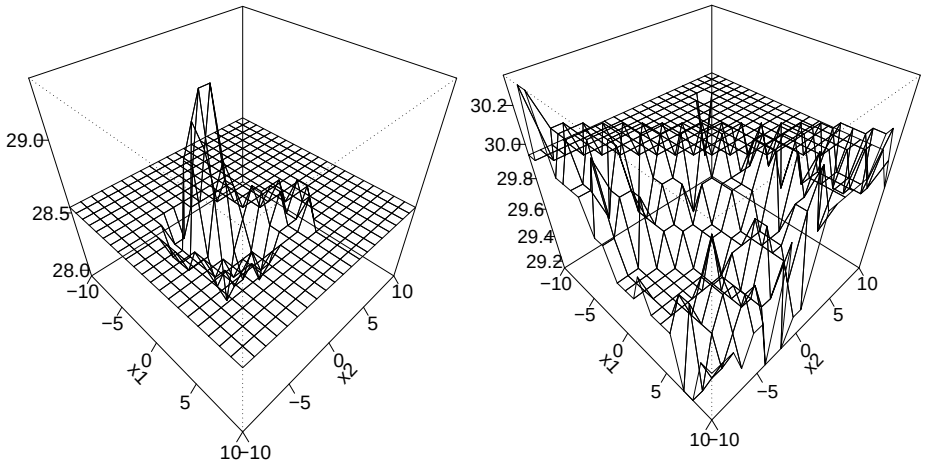


Figure 3: Percent of classification errors if one data point $z = (x, 1)$ is added to the original data set, where x varies over the grid. Left: RBF kernel. Right: linear kernel.

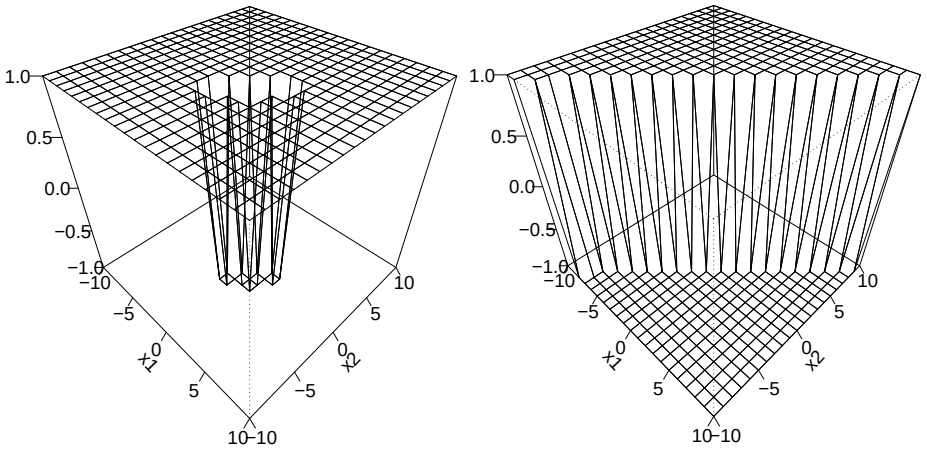


Figure 4: Fitted $y-$value for new observation if one data point $z = (x, 1)$ is added to the original data set, where x varies over the grid. Left: RBF kernel. Right: linear kernel.

It would be interesting to study the influence function of convex risk minimization methods for other problems, *e.g.* $\varepsilon-$regression or kernel principal component analysis, or to consider other robustness concepts, but this is beyond the scope of this paper.

Acknowledgments

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, "Reduction of complexity in multivariate data structures") and of DoMuS (University of Dortmund, "Model building and

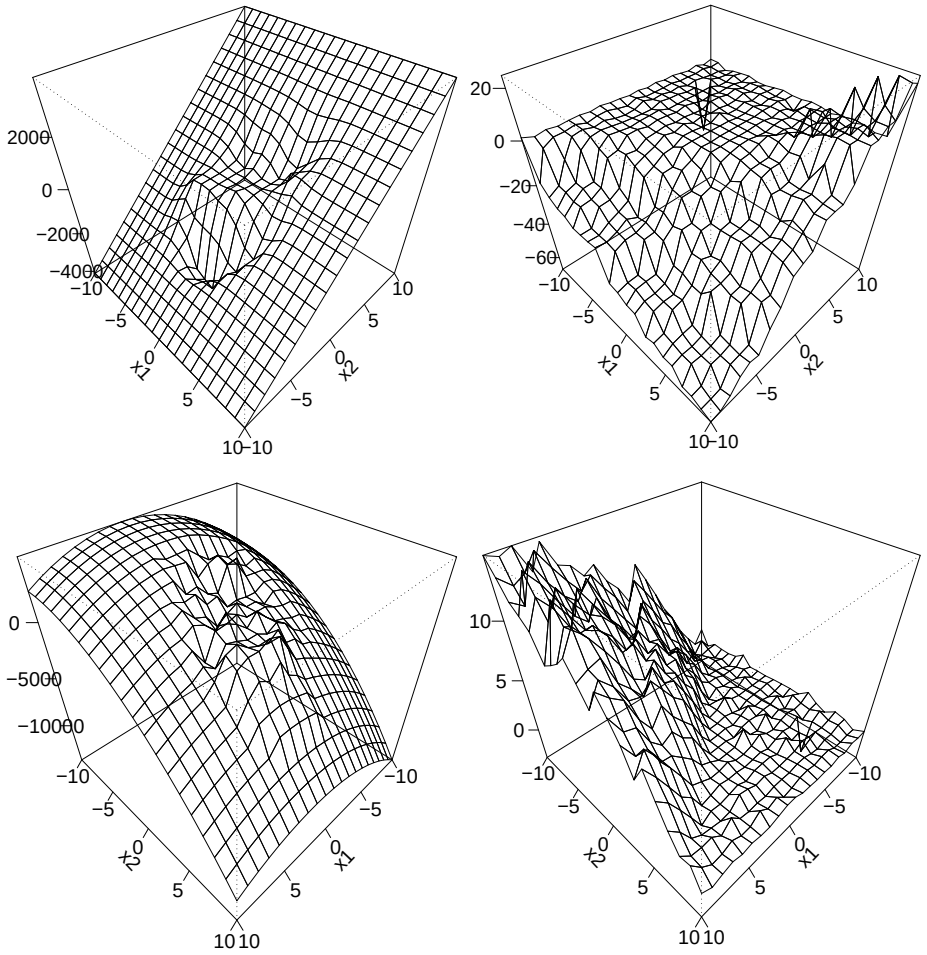


Figure 5: Sensitivity function for $\hat{\theta}$ and $\hat{b}$, respectively. Upper left: Sensitivity function for $\hat{\theta}_2$, RBF kernel. Upper right: Sensitivity function for $\hat{\theta}_2$, linear kernel. Lower left: Sensitivity function for $\hat{b}$, RBF kernel. Lower right: Sensitivity function for $\hat{b}$, linear kernel.

simulation") are gratefully acknowledged. We thank three anonymous referees and the editor for helpful remarks on an earlier version of the paper.

Appendix A. Mathematical Background

In Appendix A we list some facts from functional analysis, which are used in Appendix B to prove our theorems.

Since our proofs are heavily based on the calculus in (infinite dimensional) Banach spaces we first recall the basic facts, see *e.g.* Akerkar (1999), Brown and Pearcy (1977), and Yosida (1974). To this end let $G : E \to F$ be a map between two Banach spaces E and F. Recall, that we say that G is (Fréchet)-differentiable in $x_0 \in E$ if there exists a bounded linear operator $A : E \to F$ and a

function $\varphi : E \to F$ with $\frac{\varphi(x)}{\|x\|} \to 0$ for $x \to 0$ such that

$$G(x_0 + x) - G(x_0) \;=\; Ax + \varphi(x) \tag{16}$$

for all $x \in E$. It turns out that A is uniquely determined by (16). As in Section 4 we hence write $G'(x) := \frac{\partial G}{\partial E}(x) := A$. Again, the map G is called continuously differentiable if the map $x \mapsto G'(x)$ exists on E and is continuous. Analogously we define continuous differentiability on open subsets of E.

Unlike 1-dimensional derivates general Fréchet derivates suffer from some notational difficulties. For example, the derivate $G'(x)$ itself is a *map* for every x and thus $G'(x)$ is described by $y \mapsto G'(x)y$. Furthermore, considering partial derivates can cause notational problems too. Indeed, if *e.g.* id_E is the identity of E we have

$$x \;=\; \mathrm{id}'(x)x \;=\; \frac{\partial \mathrm{id}}{\partial E}(x)x \;=\; \frac{\partial \mathrm{id}(x)}{\partial x}x$$

where the right expression uses standard notation. We feel that the latter can cause problems for the unexperienced reader.

As in the finite dimensional case the differential operator satisfies basic calculus, that is linearity and a chain rule

$$\big(G_2 \circ G_1\big)'(x) \;=\; G_2'(G_1(x)) \circ G_1'(x)$$

for $G_1 : E_1 \to E_2$, $G_2 : E_2 \to E_3$ whenever all derivates exist in the above equation. Furthermore, for a bounded linear map $A : E \to F$ we have $A'(x) = A$ for all $x \in E$. If $G(f) := \|f\|_H^2$ for all elements f of a Hilbert space H we find $G' = 2\mathrm{id}_H$, where id_H denotes the identity on H.

Let us consider an example that helps to understand the differentiation steps in the following proofs. To this end let $\mathbb{P}$ be a probability measure on a subset $X \subset \mathbb{R}^d$ and H be a RKHS of bounded continuous functions over X with feature map $\Phi : X \to H$, *i.e.* $\Phi(x) := k(x,.)$, where k is the kernel of H. We consider the map $G : H \to \mathbb{R}$ which is defined by $Gf := \mathbb{E}_{\mathbb{P}} L \circ f$ for all $f \in H$ and a twice continuously differentiable function $L : \mathbb{R} \to \mathbb{R}$. In order to compute the derivate of this map we decompose G into $G = B \circ A \circ I$, where $I : H \to C_b(X)$ is the canonical embbeding $w \mapsto \langle w, \Phi(.)\rangle$ of H into the space of all bounded continuous functions $C_b(X)$, $A : C_b(X) \to C_b(X)$ is defined by $f \mapsto L \circ f$ and $B : C_b(X) \to \mathbb{R}$ is the functional $f \mapsto \mathbb{E}_{\mathbb{P}} f$. For the chain rule we need to compute the derivatives of these factors. Since I is linear we have $I'(v)w = Iw$ for all $v, w \in H$. Analogously, the linearity of B gives $B'(f)g = \mathbb{E}_{\mathbb{P}} g$ for all $f, g \in C_b(X)$. As shown in the book of Akerkar (1999), we also have $A'(f)g = g \cdot (L' \circ f)$ for all $f, g \in C_b(X)$, *i.e.* A' is the multiplication operator with respect to $L' \circ f$. Applying the chain rule to $A \circ I$ we hence find

$$(A \circ I)'(v)w \;=\; (A'(I(v)) \circ I'(v))w \;=\; L' \circ (Iv) \cdot (Iw)$$

for all $v, w \in H$. As we see the brackets play an important role for the mechanic evaluation of these derivatives. Another application of the chain rule gives

$$\begin{aligned} G'(v)w \;=\; \big(B'(A \circ I(v)) \circ (A \circ I)'(v)\big)w \;&=\; B'(A \circ I(v))\big((A \circ I)'(v)w\big) \\ &=\; \mathbb{E}_{\mathbb{P}}(A \circ I)'(v)w \\ &=\; \mathbb{E}_{\mathbb{P}} L' \circ (Iv) \cdot (Iw) \end{aligned}$$

for all $v, w \in H$. Note that by definition $G'(v) : H \to \mathbb{R}$ is a bounded functional. By the theorem of Fréchet-Riesz such functionals can be represented by elements of H via the mapping $v \mapsto \langle v, .\rangle$. In

our situation we can directly compute this representation: for $v, w \in H$ we have

$$\langle \mathbb{E}_{\mathbb{P}} L' \circ (Iv)\Phi, w \rangle \;=\; \mathbb{E}_{\mathbb{P}} L' \circ (Iv) \langle Iv, w \rangle \;=\; \mathbb{E}_{\mathbb{P}} L' \circ (Iv) \cdot (Iw)\,,$$

i.e. $\mathbb{E}_{\mathbb{P}} L' \circ (Iv)\Phi$ is a representation of $G'(v)$. Note that $\mathbb{E}_{\mathbb{P}} L' \circ (Iv)\Phi$ is a H-valued *Bochner-integral*. For finite dimensional spaces $\mathbb{R}^n$ the $\mathbb{R}^n$-valued Bochner-integral can be computed by the integrals of the n components. In general Banach spaces some problems can occur by different notions of measurability. Since in our case H is separable by the continuity of Φ and all our functions are continuous we do not have these difficulties. In fact for compact X our Bochner-integrals can be even computed using a simple Riemann approach. For more information about Bochner-integrals we refer to Diestel and Uhl (1977) or Yosida (1974).

Since in our proofs we also have to compute the second derivate of maps of the form of G, let us now treat G''. For convenience we use the Fréchet-Riesz representation of G'. Analogously to the above considerations we decompose G'. To this end $B : C_b(X) \to H$ denotes the operator defined by $Bf := \mathbb{E}_{\mathbb{P}} f\Phi$ for all $f \in C_b(X)$. Furthermore, $A : C_b(X) \to C_b(X)$ is the operator $Af := L' \circ f$, $f \in C_b(X)$. Using the Fréchet-Riesz representation of G' we then find $G' = B \circ A \circ I$. Now observe that B is linear. Therefore we have $B'(f)g = \mathbb{E}_{\mathbb{P}} g\Phi$ for all $g \in C_b(X)$. Moreover, we find $(A \circ I)'(v)w = L'' \circ (Iv)Iw$ for all $v, w \in H$ as above. Using the chain rule this gives

$$G''(v)w \;=\; B''(A \circ I(v))\big((A \circ I)'(v)w\big) \;=\; \mathbb{E}_{\mathbb{P}}(A \circ I)'(v)w\Phi \;=\; \mathbb{E}_{\mathbb{P}} L'' \circ (Iv)Iw\Phi$$

for all $v, w \in H$.

Our proofs also heavily rely on the implicit function theorem in Banach spaces. Therefore, we recall a simplified version of this theorem (Akerkar, 1999; Zeidler, 1986). Here and throughout this appendix B_E denotes the open unit ball of a Banach space E.

Theorem 19 (Implicit function theorem) *Let E, F be Banach spaces and $G : E \times F \to F$ be a continuously differentiable map. Suppose that we have $(x_0, y_0) \in E \times F$ such that $G(x_0, y_0) = 0$ and $\frac{\partial G}{\partial F}(x_0, y_0)$ is invertible. Then there exists a $\delta > 0$ and a continuously differentiable map $f : x_0 + \delta B_E \to y_0 + \delta B_F$ such that for all $x \in x_0 + \delta B_E$, $y \in y_0 + \delta B_F$ we have*

$$G(x, y) = 0 \qquad \text{if and only if} \qquad y = f(x)\,.$$

Moreover, the derivative of f is given by

$$f'(x) = -\left(\frac{\partial G}{\partial F}\big(x, f(x)\big)\right)^{-1} \frac{\partial G}{\partial E}\big(x, f(x)\big)\,.$$

For the application of the implicit function theorem we have to show that certain operators are invertible. For this the following theorem which is known as the Fredholm Alternative, (Cheney, 2001) turns out to be very helpful:

Theorem 20 (Fredholm Alternative) *Let E be a Banach space and $S : E \to E$ be a compact operator. Then $\mathrm{id}_E + S$ is surjective if and only if it is injective.*

We also need the Krein-Milman theorem, see Yosida (1974, p. 363) or Brown and Pearcy (1977, p. 309).

Theorem 21 (Krein-Milman theorem) *Let K be a non-void compact convex subset of a locally convex real linear topological space. Then K is equal to the closure of the convex hull of the set of all extreme points of K.*

Appendix B. Proofs of the Theorems

In this appendix we prove the theorems from Section 4 and Section 5. We sometimes write $L(f)$ instead of $L(y, f(x))$ and $L(f+b)$ instead of $L(y, f(x)+b)$ to shorten the notation, if misunderstandings are unlikely. We use this kind of notation also for derivatives of L.

PROOF OF THEOREM 4. Let us first check that the solution $f_{\mathbb{P},\lambda}$ exists in our situation. Indeed, in the proof of the existence statement by Steinwart (2002a) the compactness of X is only used to ensure $K := \|I : H \to \ell_\infty(X)\| < \infty$, where $\ell_\infty(X)$ denotes the space of all bounded functions $f : X \to \mathbb{R}$ equipped with the supremum norm and I is the canonical embbeding. The finiteness of this norm, however, *characterizes* bounded kernels. Therefore, the existence statement by Steinwart (2002a) is true in our case too.

Now, let $\Phi : X \to H$ be the feature map of H as in the above example. Our analysis heavily rely on the map $G : \mathbb{R} \times H \to H$ that is defined by

$$G(\varepsilon, f) := 2\lambda f + \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z} L'(Y, f(X))\Phi(X)\,.$$

Note that for $\varepsilon \notin [0,1]$ the H-valued expectation is with respect to a signed measure. For these measures we refer to Dudley (2002). Now as in the above example, for $\varepsilon \in [0,1]$ we obtain

$$G(\varepsilon, f) = \frac{\partial R^{reg}_{L,(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z,\lambda}}{\partial H}(f)\,. \tag{17}$$

Since $f \mapsto R^{reg}_{L,(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z,\lambda}(f)$ is convex for all $\varepsilon \in [0,1]$ Equation (17) shows that we have $G(\varepsilon, f) = 0$ if and only if $f = f_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z,\lambda}$ for such ε. Our aim is to show the existence of a differentiable function $\varepsilon \mapsto f_\varepsilon$ defined on a small interval $[-\delta,\delta]$ for some $\delta > 0$ that satisfies $G(\varepsilon, f_\varepsilon) = 0$ for all $\varepsilon \in [-\delta,\delta]$. Once we have shown the existence of this function we immediately obtain

$$IF(z; T, \mathbb{P}) = \frac{\partial f_\varepsilon}{\partial \varepsilon}(0)\,.$$

For the existence of $\varepsilon \mapsto f_\varepsilon$ we only have to check by Theorem 19 that G is continuously differentiable and that $\frac{\partial G}{\partial H}(0, f_{\mathbb{P},\lambda})$ is invertible. Let us start with the first: an easy computation shows

$$\frac{\partial G}{\partial \varepsilon}(\varepsilon, f) = -\mathbb{E}_{\mathbb{P}} L'(Y, f(X))\Phi(X) + \mathbb{E}_{\Delta_z} L'(Y, f(X))\Phi(X)\,. \tag{18}$$

Moreover, as in the above example we find

$$\frac{\partial G}{\partial H}(\varepsilon, f) = 2\lambda \mathrm{id}_H + \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z} L''(Y, f(X))\langle\Phi(X), .\rangle\Phi(X)\,. \tag{19}$$

Since H has a bounded kernel it is a simple routine to check that both partial derivatives are continuous. This together with the continuity of G ensures that G is continuously differentiable, *cf.* Akerkar (1999).

In order to show that $\frac{\partial G}{\partial H}(0, f_{\mathbb{P},\lambda})$ is invertible it suffices to show by the Fredholm Alternative that $\frac{\partial G}{\partial H}(0, f_{\mathbb{P},\lambda})$ is injective and that

$$Ag := \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X))g(X)\Phi(X)\,, \qquad g \in H,$$

defines a compact operator on H. To show the compactness we have to recall some measure theory, see Dudley (2002). Since X is assumed to be open or closed, it is a Polish space. Furthermore, Borel probability measures on Polish spaces are *regular* by Ulam's theorem, *i.e.* they can be approximated from inside by compact sets, *cf.* Bauer (1990, p. 180). In our situation, this means that for all $n \geq 1$ there exists a compact subset $X_n \subset X$ with $\mathbb{P}_X(X_n) \geq 1 - 1/n$, where $\mathbb{P}_X$ denotes the marginal distribution of $\mathbb{P}$ with respect to X. We define a sequence of operators $A_n : H \to H$ by

$$A_n g := \int_{X_n \times Y} L''(y, f_{\mathbb{P},\lambda}(x)) g(x) \Phi(x) d\mathbb{P}(x,y)$$

for all $g \in H$. Let us now show that all A_n are compact. By the definition of A_n there exists a constant $c > 0$ depending on λ, L'' and K such that for all $g \in B_H$ we have

$$A_n g \in c \cdot \overline{\operatorname{aco} \Phi(X_n)}\,, \tag{20}$$

where $\operatorname{aco}\Phi(X_n)$ denotes the absolute convex hull of $\Phi(X_n)$. Indeed, for discrete probability measures $\mathbb{P}$ relation (20) follows directly from the definition using the fact $\|g\|_\infty \leq K$ for $g \in B_H$. To see the general case recall that by Krein-Milman's theorem the set of discrete probability measures is weak*-dense in the set of probability measures. Then (20) follows since H has the approximation property (Lindenstrauss and Tzafriri, 1977). Furthermore, since Φ is continuous $\Phi(X_n)$ is compact and hence so is $\overline{\operatorname{aco}\Phi(X_n)}$. This shows that A_n is compact. In order to see that A is compact, it therefore suffices to show $\|A_n - A\| \to 0$ for $n \to \infty$. The latter convergence can be easily checked using $\mathbb{P}_X(X_n) \geq 1 - 1/n$.

It remains to prove that A is injective. To this end for $g \neq 0$ we find

$$\begin{aligned}
\big\langle (2\lambda \operatorname{id}_H + A)g, (2\lambda \operatorname{id}_H + A)g \big\rangle &= 4\lambda^2 \langle g, g\rangle + 4\lambda \langle g, Ag\rangle + \langle Ag, Ag\rangle \\
&> 4\lambda \langle g, Ag\rangle \\
&= 4\lambda \big\langle g, \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X)) g(X) \Phi(X) \big\rangle \\
&= 4\lambda \mathbb{E}_{\mathbb{P}} L''(Y, f_{\mathbb{P},\lambda}(X)) g^2(X) \\
&\geq 0
\end{aligned}$$

Here, the last equation is due to the fact that $B\mathbb{E}_{\mathbb{P}} h = \mathbb{E}_{\mathbb{P}} Bh$ for all E-valued functions h and all bounded linear operators $B : E \to F$ between Banach spaces E and F. A much stronger result is given in Diestel and Uhl (1977, p. 47). The last inequality is true since the second derivative of a convex function is always nonnegative. Obviously, the above estimate shows that $\frac{\partial G}{\partial H}(0, f_{\mathbb{P},\lambda}) = 2\lambda \operatorname{id}_H + A$ is injective.

We use $IF(z; T, \mathbb{P}) = \frac{\partial f_\varepsilon}{\partial \varepsilon}(0)$ to derive a formula for the influence function, where $\varepsilon \mapsto f_\varepsilon$ is the function implicitly defined by $G(\varepsilon, f) = 0$. The implicit function theorem hence gives

$$IF(z; T, \mathbb{P}) = -S^{-1} \circ \frac{\partial G}{\partial \varepsilon}(0, f_{\mathbb{P},\lambda})\,, \tag{21}$$

where $S := \frac{\partial G}{\partial H}(0, f_{\mathbb{P},\lambda})$. ■

PROOF OF THEOREM 6. Since every compact subset of $\mathbb{R}^d$ is closed and continuous kernels on compact subsets are bounded the assertion directly follows from Theorem 4. ■

PROOF OF THEOREM 8. The proof is similar to that of Theorem 4. However, due to the extra variable b we have to adapt our approach. As in the proof of Theorem 4 we first point out that the solutions $(f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda}) \in H \times \mathbb{R}$ exist. Again, this can be seen by a slight modification of the argument used by Steinwart (2002a). Now, let us define the map $G : \mathbb{R} \times H \times \mathbb{R} \to H \times \mathbb{R}$ by

$$G(\varepsilon, f, b) := \Big(2\lambda f + \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z} L'(f+b)\Phi, \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z} L'(f+b) \Big).$$

Again, for $\varepsilon \in [0,1]$ the definition of G ensures

$$G(\varepsilon, f, b) = \frac{\partial R^{reg}_{L,(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z,\lambda}}{\partial(H \times \mathbb{R})}(f) \,,$$

if we apply the Fréchet-Riesz identification $(H \times \mathbb{R})' = H \times \mathbb{R}$. Since $R^{reg}_{L,(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z,\lambda}$ is convex for all $\varepsilon \in [0,1]$ we have $G(\varepsilon, f, b) = 0$ if and only if (f, b) minimizes $R^{reg}_{L,(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z,\lambda}$ for such ε. Our aim is to apply the implicit function theorem in the way we did it in the proof of Theorem 4. However, this time the implicit function theorem will also ensure the uniqueness of the solution of (7). Obviously, this is necessary for the existence of the influence function. In order to apply Theorem 19 we need the partial derivatives of G. By an easy computation we find

$$\frac{\partial G}{\partial \varepsilon}(\varepsilon, f, b) = -\mathbb{E}_{\mathbb{P}} L'(f+b)\Phi(X) + \mathbb{E}_{\Delta_z} L'(f+b)\Phi(X)$$

and

$$\frac{\partial G}{\partial(H \times \mathbb{R})}(\varepsilon, f, b) = \begin{pmatrix} 2\lambda \mathrm{id}_H + \mathbb{E}_\varepsilon L''(f+b)\langle \Phi, . \rangle \Phi & \mathbb{E}_\varepsilon L''(f+b)\Phi \\ \mathbb{E}_\varepsilon L''(f+b)\Phi & \mathbb{E}_\varepsilon L''(f+b) \end{pmatrix},$$

where we use the abbreviation $\mathbb{E}_\varepsilon := \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\Delta_z}$. A routine check shows that both G and the partial derivatives are continuous and hence G is continuously differentiable.
Now, let us fix a solution $(f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$ of (7). In order to show that the operator $\frac{\partial G}{\partial(H\times\mathbb{R})}(0,\, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$ is invertible it suffices to show by the Fredholm Alternative that it is injective and that

$$A := \begin{pmatrix} \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda})\langle \Phi, . \rangle \Phi & \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda})\Phi \\ \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda})\Phi & \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda}) - 2\lambda \end{pmatrix}$$

is a compact operator on $H \times \mathbb{R}$. The latter can be seen using the argument of the proof of Theorem 4. For the former let us suppose that we have an element $(g,t) \in H \times \mathbb{R}$ with $(2\lambda \mathrm{id}_{H\times\mathbb{R}} + A)(g,t) = 0$. This is equivalent to the following linear system of equations

$$\begin{aligned} 2\lambda g + \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda}) g \Phi + t\, \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda})\Phi &= 0 \qquad (22) \\ \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda}) g + t\, \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda}) &= 0\,. \qquad (23) \end{aligned}$$

Let us first assume that $t = 0$. Then the above system yields

$$2\lambda g + \mathbb{E}_{\mathbb{P}} L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda}) g \Phi = 0\,.$$

Using the techniques of the proof of Theorem 4 we easily find that this implies $g = 0$. Therefore, we may assume without loss of generality that $t = 1$. In order to avoid long notations we introduce the measure $d\mu := L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda}) d\mathbb{P}$. Note that $L'' > 0$ implies $\mu \neq 0$. Now, (23) yields

$$\mu(g) = -\mu(1)\,, \qquad (24)$$

where 1 denotes the constant function with value 1. Hence, by (22) we find

$$0 = 2\lambda\langle g,g\rangle + \mu(g^2) + \mu(g) = 2\lambda\langle g,g\rangle + \mu(g^2) - \mu(1)\ . \tag{25}$$

Furthermore, (24) implies

$$0 \le \mu((g+1)^2) = \mu(g^2) + 2\mu(g) + \mu(1) = \mu(g^2) - \mu(1)\,.$$

This together with (25) yields $2\lambda\langle g,g\rangle \le 0$ and hence $g = 0$. However, the latter contradicts (24) and hence there is no non-trivial solution of the system (22), (23).
Now, the implicit function theorem states in particular, that the solution $(f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$ is unique in a small neighborhood of $(f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$. Hence it is globally unique since the set of solutions of (7) is convex. The rest of the proof follows the ideas of the proof of Theorem 4.

We use $IF(z;T,\mathbb{P}) = \frac{\partial(f_\varepsilon, b_\varepsilon)}{\partial\varepsilon}(0)$ to derive a formula for the influence function, where $\varepsilon \mapsto (f_\varepsilon, b_\varepsilon)$ is the function implicitly defined by $G(\varepsilon, f, b) = 0$. The implicit function theorem hence gives

$$IF(z;T,\mathbb{P}) = -S^{-1} \circ \frac{\partial G}{\partial\varepsilon}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})\ , \tag{26}$$

where $S := \frac{\partial G}{\partial(H\times\mathbb{R})}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$. ■

PROOF OF THEOREM 12. Recall that every convex function on $\mathbb{R}$ is locally Lipschitz continuous. Let $|L_{|Y\times[-c,c]}|_1$ denote the Lipschitz constant of L restricted to $Y \times [-c,c]$, $c > 0$. We define $\delta_\lambda := \sqrt{(L(-1,0) + L(1,0))/\lambda}$ and $K := \sup_{x\in X}\sqrt{k(x,x)}$. We fix a distribution $\mathbb{P}$. An easy estimate (Steinwart, 2002a) shows $\left\| f_{\mathbb{P},\lambda}\right\|_\infty \le \delta_\lambda K$. We will apply Theorem 28 in Steinwart (2003). Since this result is only formulated for compact subsets X we first have to check that it is also true in our situation. Indeed Propositions 26 and 27 in Steinwart (2003) only uses the boundedness of the kernel. Furthermore, the proof of Theorem 28 itself only uses the boundedness, too. Hence this theorem actually holds in our case! Thus, there exists a measurable function $h : X \times Y \to \mathbb{R}$ with $\|h\|_\infty \le \left|L_{|Y\times[-\delta_\lambda K, \delta_\lambda K]}\right|_1$ such that for all distributions $\hat{\mathbb{P}}$ we have

$$\|f_{\mathbb{P},\lambda} - f_{\hat{\mathbb{P}},\lambda}\|_H \le \frac{1}{\lambda}\left\|\mathbb{E}_{\mathbb{P}} h\Phi - \mathbb{E}_{\hat{\mathbb{P}}} h\Phi\right\|_H,$$

where $\Phi : X \to H$ is the feature map of H. Let $\hat{\mathbb{P}} := (1-\varepsilon)\mathbb{P} + \varepsilon\tilde{\mathbb{P}}$. Then the above inequality yields

$$\begin{aligned}
\varepsilon^{-1}\|f_{(1-\varepsilon)\mathbb{P}+\varepsilon\tilde{\mathbb{P}},\lambda} - f_{\mathbb{P},\lambda}\|_H &\le (\varepsilon\lambda)^{-1}\left\|\mathbb{E}_{\mathbb{P}} h\Phi - \mathbb{E}_{(1-\varepsilon)\mathbb{P}+\varepsilon\tilde{\mathbb{P}}} h\Phi\right\|_H \\
&= \lambda^{-1}\left\|\mathbb{E}_{\mathbb{P}} h\Phi - \mathbb{E}_{\tilde{\mathbb{P}}} h\Phi\right\|_H \\
&\le c_L(\lambda)\,\|\mathbb{P} - \tilde{\mathbb{P}}\|_{\mathcal{M}}\,,
\end{aligned}$$

where

$$c_L(\lambda) = \lambda^{-1} K\left|L_{|Y\times[-\delta_\lambda K, \delta_\lambda K]}\right|_1. \tag{27}$$

This shows the assertion. ■

For the proof of Theorem 15 we need the following result.

Lemma 22 *Let $\lambda, \mu \in (0,\infty)$ be fixed constants. Then the function $g : (0,1) \to \mathbb{R}$, $g(s) = 2\lambda s^2 - 2\mu s[1-s^2]^{1/2} + \mu$ has a global minimum in the point*

$$s_{min} = 2^{-1/2}\left(1 - \frac{\lambda}{(\mu^2+\lambda^2)^{1/2}}\right)^{1/2}$$

and

$$g(s_{min}) = \lambda\left(1 - \frac{\lambda}{(\mu^2+\lambda^2)^{1/2}}\right) + \mu\left(1 - \left(1 - \frac{\lambda^2}{\mu^2+\lambda^2}\right)^{1/2}\right) > 0\,.$$

PROOF OF LEMMA 22. Let $s \in (0,1)$. We have $g'(s) = 4\lambda s - 2\mu(1-2s^2)[1-s^2]^{-1/2}$ and

$$g''(s) = 4\lambda + \left(4s - \frac{s(1-2s^2)}{1-s^2}\right)\frac{2\mu}{\sqrt{1-s^2}} \geq 4\lambda + 3s\frac{2\mu}{\sqrt{1-s^2}} > 0$$

for all $s \in (0,1)$. Define $a := \lambda/\left[\mu^2+\lambda^2\right]^{1/2}$. We have $4\lambda s_{min} > 0$ and

$$\begin{aligned}\frac{2\mu(1-2s_{min}^2)[1-s_{min}^2]^{-1/2}}{4\lambda s_{min}} &= \frac{\mu}{\lambda}\frac{a}{[(1+a)(1-a)]^{1/2}} \\ = \frac{\mu}{\lambda}\left[a^{-2}-1\right]^{-1/2} &= \frac{\mu}{\lambda}\left[\frac{\mu^2+\lambda^2}{\lambda^2} - 1\right]^{-1/2} = 1\,.\end{aligned}$$

This yields $g'(s_{min}) = 0$ and we obtain the expression $g(s_{min}) = \lambda\left[1 - \frac{\lambda}{(\mu^2+\lambda^2)^{1/2}}\right] + \mu\left[1 - (1 - \frac{\lambda^2}{\mu^2+\lambda^2})^{1/2}\right]$. ■

PROOF OF THEOREM 15. By rescaling problem (7) we may assume without loss of generality that $K := \sup_{x\in X}\sqrt{k(x,x)} \leq 1$. Recall, that in the proof of Theorem 8 we used

$$IF(z;T,\mathbb{P}) = \frac{\partial(f_\varepsilon, b_\varepsilon)}{\partial\varepsilon}(0)\,,$$

where $\varepsilon \mapsto (f_\varepsilon, b_\varepsilon)$ was the function implicitly defined by $G(\varepsilon, f, b) = 0$. The implicit function theorem hence gives

$$IF(z;T,\mathbb{P}) = -S^{-1} \circ \frac{\partial G}{\partial\varepsilon}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})\,, \tag{28}$$

where $S := \frac{\partial G}{\partial(H\times\mathbb{R})}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})$. Therefore, it suffices to bound the norms of the operators on the right side of (28). We begin with

$$\begin{aligned}\left\|\frac{\partial G}{\partial\varepsilon}(0, f_{\mathbb{P},\lambda}, b_{\mathbb{P},\lambda})\right\| &= \|\mathbb{E}_{\mathbb{P}}L'(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\Phi - \mathbb{E}_{\Delta_z}L'(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\Phi\| \\ &\leq b_{\mathbb{P},\lambda}\,\|\mathbb{P}-\Delta_z\|_{\mathcal{M}}\,.\end{aligned}$$

Furthermore, for $(g,t) \in H\times\mathbb{R}$ we have

$$\begin{aligned}S(g,t) &= \begin{pmatrix} 2\lambda\,\mathrm{id}_H + \mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\langle\Phi,.\rangle\Phi & \mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\Phi \\ \mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\Phi & \mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\end{pmatrix}\begin{pmatrix} g \\ t\end{pmatrix} \\ &= \begin{pmatrix} 2\lambda g + \mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})g\Phi + t\mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\Phi \\ \mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})g + t\mathbb{E}_{\mathbb{P}}L''(f_{\mathbb{P},\lambda}+b_{\mathbb{P},\lambda})\end{pmatrix}.\end{aligned}$$

As in the proof of Theorem 8 we write $d\mu := L''(f_{\mathbb{P},\lambda} + b_{\mathbb{P},\lambda})d\mathbb{P}$. Then we find

$$\langle S(g,t),(g,t)\rangle = 2\lambda\langle g,g\rangle + \mu(g^2) + 2t\mu(g) + t^2\mu(1)\,. \tag{29}$$

Let us suppose that $\|(g,t)\| = 1$. Then there exist $w \in H$ with $\|w\| = 1$ and $s \in [0,1]$ such that $g = sw$ and $t = \pm\sqrt{1-s^2}$. If $\mu(w) \geq 0$ Equation (29) then yields

$$\langle S(g,t),(g,t)\rangle \;\geq\; 2\lambda s^2 + \frac{s^2\mu^2(w)}{\mu(1)} - 2s\sqrt{1-s^2}\mu(w) + (1-s^2)\mu(1)\,. \tag{30}$$

Here, we used $\mu(w) \leq \sqrt{\mu(1)}\sqrt{\mu(w)}$. It is easy to check that (30) also holds if $\mu(w) \leq 0$. Now, recall that by the assumption of the theorem we have $L'' \geq a$. This implies $\mu(1) \geq a > 0$. For the special case $s = 0$ it follows from (30), that

$$\langle S(g,t),(g,t)\rangle \geq \mu(1) > a > 0. \tag{31}$$

For the special case $s = 1$ it follows from (30), that

$$\langle S(g,t),(g,t)\rangle \geq 2\lambda + \frac{\mu^2(w)}{\mu(1)} \geq 2\lambda > 0\,. \tag{32}$$

Now, we will consider the case $s \in (0,1)$. We first minimize the right hand side of (30) with respect to $\mu(w)$. To this end recall that $K \leq 1$ implies $|\mu(w)| \leq \mu(1)$. Therefore, the function $\mu(w) \mapsto \frac{s^2\mu^2(w)}{\mu(1)} - 2s\sqrt{1-s^2}\mu(w)$ is minimal if

$$\mu(w) = \min\{\mu(1), \mu(1)s^{-1}\sqrt{1-s^2}\}\,. \tag{33}$$

Using (30) it follows for $\mu(w) = \mu(1)$ that

$$\langle S(g,t),(g,t)\rangle \;\geq\; 2\lambda s^2 + \mu(1)\big(1 - 2s\sqrt{1-s^2}\big) = 2\lambda s^2 - 2\mu(1)s\sqrt{1-s^2} + \mu(1)\,. \tag{34}$$

Hence, Lemma 22 and (34) yield for the case $s \in (0,1)$ and $\mu(w) = \mu(1)$ that

$$\langle S(g,t),(g,t)\rangle \;\geq\; c_{low} \in (0,\infty)\,, \tag{35}$$

where

$$c_{low} = \lambda\left(1 - \frac{\lambda}{(\mu^2(1)+\lambda^2)^{1/2}}\right) + \mu(1)\left(1 - \left(1 - \frac{\lambda^2}{\mu^2(1)+\lambda^2}\right)^{1/2}\right).$$

By (33) we finally have to treat the case $s \in (0,1)$ and $\mu(w) = \mu(1)s^{-1}\sqrt{1-s^2}$. In this case we have $s^2 \geq \frac{1}{2}$ by $\mu(w) \leq \mu(1)$. Hence we obtain from (30) that

$$\langle S(g,t),(g,t)\rangle \;\geq\; 2\lambda s^2 + (1-s^2)\mu(1) - 2(1-s^2)\mu(1) + (1-s^2)\mu(1) \;=\; 2\lambda s^2 \;\geq\; \lambda\,. \tag{36}$$

Combining (31), (32), (35), and (36) we obtain

$$\langle S(g,t),(g,t)\rangle \;\geq\; \min\{a, 2\lambda, c_{low}, \lambda\} > 0\,. \tag{37}$$

Therefore, by the proof of Pedersen (1989, Prop. 3.2.12) it follows

$$\|S(g,t)\| \geq \min\{a,\lambda,c_{low}\}\,\|(g,t)\|$$

for all $(g,t) \in H \times \mathbb{R}$, and

$$\|S^{-1}\| \leq \frac{1}{\min\{a,\lambda,c_{low}\}} \in (0,\infty)\,.$$

This shows the assertion. ■

References

R. Akerkar. *Nonlinear Functional Analysis*. Narosa Publishing House, New Dehli, 1999.

P. L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. `http://www.stat.berkeley.edu/~bartlett/papers/bbm-lrc-02b.pdf`, 2002.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. `http://stat-www.berkeley.edu/tech-reports/638.pdf`, 2003.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

H. Bauer. *Maß- und Integrationstheorie*. De Gruyter, Berlin, 1990.

G. Blanchard, G. Lugosi, and N. Vayatis. On the rate of convergence of regularized boosting classifiers. *Journal of Machine Learning Research*, 4:861–894, 2003.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

A. Brown and C. Pearcy. *Introduction to Operator Theory I*. Springer, New York, 1977.

D.-R. Chen, Q. Wu, Y. Ying, and D.-X. Zhou. Support vector machine soft margin classifiers. City University of Hong Kong, 2003.

W. Cheney. *Analysis for Applied Mathematics*. Springer, New York, 2001.

A. Christmann. Least median of weighted squares in logistic regression with large strata. *Biometrika*, 81:413–417, 1994.

A. Christmann. *On positive breakdown point estimators in regression models with discrete response variables*. 1998. Habilitation thesis, University of Dortmund, Department of Statistics.

A. Christmann. On a strategy to develop robust and simple tariffs from motor vehicle insurance data. Technical Report 16/04, University of Dortmund, SFB-475, 2004. `http://www.statistik.uni-dortmund.de/download/mitarbeiter/christmann/Christmann-insurance04.pdf`.

A. Christmann, P. Fischer, and T. Joachims. Comparison between various regression depth methods and the support vector machine to approximate the minimum number of misclassifications. *Computational Statistics*, 17:273–287, 2002.

A. Christmann and P. J. Rousseeuw. Measuring overlap in logistic regression. *Computational Statistics and Data Analysis*, 37:65–75, 2001.

J. Diestel and J. J. Uhl. *Vector Measures*. American Mathematical Society, Providence, RI, 1977.

D. L. Donoho and P. J. Huber. The notion of breakdown point. In P. J. Bickel, K. A. Doksum, and J. L. Hodges Jr, editors, *A Festschrift for Erich L. Lehmann*, pages 157–184, Belmont, California, Wadsworth, 1983.

R. M. Dudley. *Real Analysis and Probability*. Cambridge University Press, 2002.

Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the 13th International Conference*, pages 148–156. Morgan Kauffman Publishers, San Francisco, 1996.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *Annals of Statistics*, 28:337–407, 2000.

F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69:383–393, 1974.

F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics. The approach based on influence functions*. Wiley, New York, 1986.

D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, Cambridge, Massachusetts, 2001.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

J. Hipp, U. Güntzer, and U. Grimmer. Data quality mining - making a virtue of necessity. Workshop on Research Issues in Data Mining and Knowledge Discovery DMKD, Santa Barbara, CA, `http://www.cs.cornell.edu/johannes/papers/dmkd2001-papers/p5_hipp.pdf`, 2001.

K. U. Höffgen, H.-U. Simon, and K. S. van Horn. Robust trainability of single neurons. *Journal Computer and System Sciences*, 50:114–125, 1995.

P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35: 73–101, 1964.

P. J. Huber. *Robust statistics*. Wiley, New York, 1981.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 41–56, MIT Press, Cambridge, Massachusetts, 1999.

J. Lindenstrauss and L. Tzafriri. *Classicial Banach spaces I*. Springer, Berlin, 1977.

G. K. Pedersen. *Analysis Now*. Springer, New York, 1989.

H. Rieder. *Robust Asymptotic Statistics*. Springer, New York, 1994.

P. J. Rousseeuw and M. Hubert. Regression depth. *Journal of the American Statistical Association*, 94:388–433, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Massachusetts, 2002.

J. C. Scovel and I. Steinwart. Fast rates for support vector machines. Los Alamos Technical Report LA-UR-03-9117, `http://www.c3.lanl.gov/~ingo/publications/ann-03.ps`, 2003.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

I. Steinwart. Consistency of support vector machines and other regularized kernel machine. *IEEE Transactions on Information Theory*, to appear, 2002a.

I. Steinwart. Support vector machines are universally consistent. *Journal of Complexity*, 18:768–791, 2002b.

I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4: 1071–1105, 2003.

I. Steinwart. Sparseness of support vector machines – some asymptotically sharp bounds. *Proceedings of Neural Information Processing Systems*, in press, 2004.

J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48:85–105, 2002.

A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32: 135–166, 2004.

J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Reading, Massachusetts, 1977.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In B. Schölkopf, C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 69–88, MIT Press, Cambridge, Massachusetts, 1999.

K. Yosida. *Functional Analysis*. Springer, Berlin, 4^{th} edition, 1974.

E. Zeidler. *Nonlinear Functional Analysis and its Applications I*. Springer, New York, 1986.

T. Zhang. Statistical behaviour and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–134, 2004.

Journal of Machine Learning Research 5 (2004) 1035-1062 Submitted 1/04; Published 8/04

Rational Kernels: Theory and Algorithms

Corinna Cortes CORINNA@GOOGLE.COM
Google Labs
1440 Broadway
New York, NY 10018, USA

Patrick Haffner HAFFNER@RESEARCH.ATT.COM
AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932, USA

Mehryar Mohri MOHRI@RESEARCH.ATT.COM
AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932, USA

Editors: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

Many classification algorithms were originally designed for fixed-size vectors. Recent applications in text and speech processing and computational biology require however the analysis of variable-length sequences and more generally weighted automata. An approach widely used in statistical learning techniques such as Support Vector Machines (SVMs) is that of kernel methods, due to their computational efficiency in high-dimensional feature spaces. We introduce a general family of kernels based on weighted transducers or rational relations, *rational kernels*, that extend kernel methods to the analysis of variable-length sequences or more generally weighted automata. We show that rational kernels can be computed efficiently using a general algorithm of composition of weighted transducers and a general single-source shortest-distance algorithm.

Not all rational kernels are *positive definite and symmetric* (PDS), or equivalently verify the Mercer condition, a condition that guarantees the convergence of training for discriminant classification algorithms such as SVMs. We present several theoretical results related to PDS rational kernels. We show that under some general conditions these kernels are closed under sum, product, or Kleene-closure and give a general method for constructing a PDS rational kernel from an arbitrary transducer defined on some non-idempotent semirings. We give the proof of several characterization results that can be used to guide the design of PDS rational kernels. We also show that some commonly used string kernels or similarity measures such as the edit-distance, the convolution kernels of Haussler, and some string kernels used in the context of computational biology are specific instances of rational kernels. Our results include the proof that the edit-distance over a non-trivial alphabet is not *negative definite*, which, to the best of our knowledge, was never stated or proved before.

Rational kernels can be combined with SVMs to form efficient and powerful techniques for a variety of classification tasks in text and speech processing, or computational biology. We describe examples of general families of PDS rational kernels that are useful in many of these applications and report the result of our experiments illustrating the use of rational kernels in several difficult large-vocabulary spoken-dialog classification tasks based on deployed spoken-dialog systems. Our

results show that rational kernels are easy to design and implement and lead to substantial improvements of the classification accuracy.

1. Introduction

Many classification algorithms were originally designed for fixed-length vectors. Recent applications in text and speech processing and computational biology require however the analysis of variable-length sequences and more generally weighted automata. Indeed, the output of a large-vocabulary speech recognizer for a particular input speech utterance, or that of a complex information extraction system combining several knowledge sources for a specific input query, is typically a weighted automaton compactly representing a large set of alternative sequences. The weights assigned by the system to each sequence are used to rank different alternatives according to the models the system is based on. The error rate of such complex systems is still too high in many tasks to rely only on their one-best output, thus it is preferable instead to use the full weighted automata which contain the correct result in most cases.

An approach widely used in statistical learning techniques such as Support Vector Machines (SVMs) (Boser et al., 1992; Cortes and Vapnik, 1995; Vapnik, 1998) is that of kernel methods, due to their computational efficiency in high-dimensional feature spaces. We introduce a general family of kernels based on weighted transducers or rational relations, *rational kernels*, that extend kernel methods to the analysis of variable-length sequences or more generally weighted automata.[1] We show that rational kernels can be computed efficiently using a general algorithm of composition of weighted transducers and a general single-source shortest-distance algorithm.

Not all rational kernels are *positive definite and symmetric* (PDS), or equivalently verify the Mercer condition (Berg et al., 1984), a condition that guarantees the convergence of training for discriminant classification algorithms such as SVMs. We present several theoretical results related to PDS rational kernels. We show that under some general conditions these kernels are closed under sum, product, or Kleene-closure and give a general method for constructing a PDS rational kernel from an arbitrary transducer defined on some non-idempotent semirings. We give the proof of several characterization results that can be used to guide the design of PDS rational kernels.

We also study the relationship between rational kernels and some commonly used string kernels or similarity measures such as the edit-distance, the convolution kernels of Haussler (Haussler, 1999), and some string kernels used in the context of computational biology (Leslie et al., 2003). We show that these kernels are all specific instances of rational kernels. In each case, we explicitly describe the corresponding weighted transducer. These transducers are often simple and efficient for computing kernels. Their diagram provides more insight into the definition of kernels and can guide the design of new kernels. Our results also include the proof of the fact that the edit-distance over a non-trivial alphabet is not *negative definite*, which, to the best of our knowledge, was never stated or proved before.

Rational kernels can be combined with SVMs to form efficient and powerful techniques for a variety of applications to text and speech processing, or to computational biology. We describe examples of general families of PDS rational kernels that are useful in many of these applications. We report the result of our experiments illustrating the use of rational kernels in several difficult large-vocabulary spoken-dialog classification tasks based on deployed spoken-dialog systems. Our results

1. We have described in shorter publications part of the material presented here (Cortes et al., 2003a,b,c,d).

SEMIRING	SET	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{0,1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R}\cup\{-\infty,+\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R}\cup\{-\infty,+\infty\}$	min	$+$	$+\infty$	0

Table 1: Semiring examples. $\oplus_{\log}$ is defined by $x \oplus_{\log} y = -\log(e^{-x}+e^{-y})$.

show that rational kernels are easy to design and implement and lead to substantial improvements of the classification accuracy.

The paper is organized as follows. In the following section, we introduce the notation and some preliminary algebraic and automata-theoretic definitions used in the remaining sections. Section 3 introduces the definition of rational kernels. In Section 4, we present general algorithms that can be used to compute rational kernels efficiently. Section 5 introduces the classical definitions of positive definite and negative definite kernels and gives a number of novel theoretical results, including the proof of some general closure properties of PDS rational kernels, a general construction of PDS rational kernels starting from an arbitrary weighted transducer, a characterization of acyclic PDS rational kernels, and the proof of the closure properties of a very general class of PDS rational kernels. Section 6 studies the relationship between some commonly used kernels and rational kernels. Finally, the results of our experiments in several spoken-dialog classification tasks are reported in Section 7.

2. Preliminaries

In this section, we present the algebraic definitions and notation needed to introduce rational kernels.

A system $(\mathbb{K},\odot,e)$ is a *monoid* if it is closed under $\odot$: $a\odot b\in\mathbb{K}$ for all $a,b\in\mathbb{K}$; $\odot$ is associative: $(a\odot b)\odot c = a\odot(b\odot c)$ for all $a,b,c\in\mathbb{K}$; and e is an identity for $\odot$: $a\odot e = e\odot a = a$, for all $a\in\mathbb{K}$. When additionally $\odot$ is commutative: $a\odot b = b\odot a$ for all $a,b\in\mathbb{K}$, then $(\mathbb{K},\odot,e)$ is said to be a *commutative monoid*.

Definition 1 (Kuich and Salomaa (1986)) *A system $(\mathbb{K},\oplus,\otimes,\overline{0},\overline{1})$ is a* semiring *if: $(\mathbb{K},\oplus,\overline{0})$ is a commutative monoid with identity element $\overline{0}$; $(\mathbb{K},\otimes,\overline{1})$ is a monoid with identity element $\overline{1}$; $\otimes$ distributes over $\oplus$; and $\overline{0}$ is an annihilator for $\otimes$: for all $a\in\mathbb{K}, a\otimes\overline{0}=\overline{0}\otimes a=\overline{0}$.*

Thus, a semiring is a ring that may lack negation. Table 1 lists some familiar semirings. In addition to the Boolean semiring and the probability semiring, two semirings often used in applications are the *log semiring* which is isomorphic to the probability semiring via a log morphism, and the *tropical semiring* which is derived from the log semiring using the Viterbi approximation.

Definition 2 *A* weighted finite-state transducer *T over a semiring $\mathbb{K}$ is an 8-tuple $T=(\Sigma,\Delta,Q,I,F,E,\lambda,\rho)$ where Σ is the finite input alphabet of the transducer; Δ is the finite output alphabet; Q is a finite set of states; $I\subseteq Q$ the set of initial states; $F\subseteq Q$ the set of final states; $E\subseteq Q\times(\Sigma\cup\{\varepsilon\})\times(\Delta\cup\{\varepsilon\})\times\mathbb{K}\times Q$ a finite set of transitions; $\lambda: I\to\mathbb{K}$ the initial weight function; and $\rho: F\to\mathbb{K}$ the final weight function mapping F to $\mathbb{K}$.*

Weighted automata can be formally defined in a similar way by simply omitting the input or output labels.

Given a transition $e \in E$, we denote by $p[e]$ its origin or previous state and $n[e]$ its destination state or next state, and $w[e]$ its weight. A *path* $\pi = e_1 \cdots e_k$ is an element of E^* with consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. We extend n and p to paths by setting $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. A *cycle* π is a path whose origin and destination coincide: $p[\pi] = n[\pi]$. A weighted automaton or transducer is said to be *acyclic* if it admits no cycle. A *successful path* in a weighted automaton or transducer M is a path from an initial state to a final state. The weight function w can also be extended to paths by defining the weight of a path as the $\otimes$-product of the weights of its constituent transitions: $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. We denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, y, q')$ the set of paths from q to q' with input label $x \in \Sigma^*$ and output label $y \in \Delta^*$. These definitions can be extended to subsets $R, R' \subseteq Q$, by $P(R, x, y, R') = \cup_{q \in R, q' \in R'} P(q, x, y, q')$. We denote by $w[M]$ the $\oplus$-sum of the weights of all the successful paths of the automaton or transducer M, when that sum is well-defined and in $\mathbb{K}$. A transducer T is *regulated* if the output weight associated by T to any pair of input-output string (x, y) by

$$[\![T]\!](x, y) = \bigoplus_{\pi \in P(I, x, y, F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho[n[\pi]]$$

is well-defined and in $\mathbb{K}$. $[\![T]\!](x, y) = \overline{0}$ when $P(I, x, y, F) = \emptyset$. If for all $q \in Q$, the sum $\bigoplus_{\pi \in P(q, \varepsilon, \varepsilon, q)} w[\pi]$ is in $\mathbb{K}$, then T is regulated. In particular, when T does not have any ε-cycle, that is a cycle labeled with ε (both input and output labels), it is regulated. In the following, we will assume that all the transducers considered are regulated. Regulated weighted transducers are closed under the rational operations: $\oplus$-sum, $\otimes$-product and Kleene-closure which are defined as follows for all transducers T_1 and T_2 and $(x, y) \in \Sigma^* \times \Delta^*$:

$$\begin{aligned}
[\![T_1 \oplus T_2]\!](x, y) &= [\![T_1]\!](x, y) \oplus [\![T_2]\!](x, y), \\
[\![T_1 \otimes T_2]\!](x, y) &= \bigoplus_{x = x_1 x_2, y = y_1 y_2} [\![T_1]\!](x_1, y_1) \otimes [\![T_2]\!](x_2, y_2), \\
[\![T^*]\!](x, y) &= \bigoplus_{n=0}^{\infty} T^n(x, y),
\end{aligned}$$

where T^n stands for the $(n-1)$-$\otimes$-product of T with itself.

For any transducer T, we denote by T^{-1} its *inverse*, that is the transducer obtained from T by transposing the input and output labels of each transition and the input and output alphabets.

Composition is a fundamental operation on weighted transducers that can be used in many applications to create complex weighted transducers from simpler ones. Let $T_1 = (\Sigma, \Delta, Q_1, I_1, F_1, E_1, \lambda_1, \rho_1)$ and $T_2 = (\Delta, \Omega, Q_2, I_2, F_2, E_2, \lambda_2, \rho_2)$ be two weighted transducers defined over a commutative semiring $\mathbb{K}$ such that Δ, the output alphabet of T_1, coincides with the input alphabet of T_2. Then, the result of the composition of T_1 and T_2 is a weighted transducer $T_1 \circ T_2$ which, when it is regulated, is defined for all x, y by (Berstel, 1979; Eilenberg, 1974; Salomaa and Soittola, 1978; Kuich and Salomaa, 1986)[2]

$$[\![T_1 \circ T_2]\!](x, y) = \bigoplus_{z \in \Delta^*} [\![T_1]\!](x, z) \otimes [\![T_2]\!](z, y).$$

2. We use a *matrix notation* for the definition of composition as opposed to a *functional notation*.

Note that a transducer can be viewed as a matrix over a countable set $\Sigma^* \times \Delta^*$ and composition as the corresponding matrix-multiplication.

The definition of composition extends naturally to weighted automata since a weighted automaton can be viewed as a weighted transducer with identical input and output labels for each transition. The corresponding transducer associates $[\![A]\!](x)$ to a pair (x,x), and 0 to all other pairs. Thus, the composition of a weighted automaton $A_1 = (\Delta, Q_1, I_1, F_1, E_1, \lambda_1, \rho_1)$ and a weighted transducer $T_2 = (\Delta, \Omega, Q_2, I_2, F_2, E_2, \lambda_2, \rho_2)$ is simply defined for all x, y in $\Delta^* \times \Omega^*$ by

$$[\![A_1 \circ T_2]\!](x,y) = \bigoplus_{x \in \Delta^*} [\![A_1]\!](x) \otimes [\![T_2]\!](x,y)$$

when these sums are well-defined and in $\mathbb{K}$. *Intersection* of two weighted automata is the special case of composition where both operands are weighted automata, or equivalently weighted transducers with identical input and output labels for each transition.

3. Definitions

Let X and Y be non-empty sets. A function $K : X \times Y \to \mathbb{R}$ is said to be a *kernel* over $X \times Y$. This section introduces *rational kernels*, which are kernels defined over sets of strings or weighted automata.

Definition 3 *A kernel K over $\Sigma^* \times \Delta^*$ is said to be* rational *if there exist a weighted transducer $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ over the semiring $\mathbb{K}$ and a function $\psi : \mathbb{K} \to \mathbb{R}$ such that for all $x \in \Sigma^*$ and $y \in \Delta^*$:*[3]

$$K(x,y) = \psi([\![T]\!](x,y)).$$

K is then said to be defined by the pair (ψ, T).

This definition and many of the results presented in this paper can be generalized by replacing the free monoids Σ^* and Δ^* with arbitrary monoids M_1 and M_2. Also, note that we are not making any particular assumption about the function ψ in this definition. In general, it is an arbitrary function mapping $\mathbb{K}$ to $\mathbb{R}$.

Figure 1 shows an example of a weighted transducer over the probability semiring corresponding to the gappy n-gram kernel with decay factor λ as defined by (Lodhi et al., 2001). Such gappy n-gram kernels are rational kernels (Cortes et al., 2003c).

Rational kernels can be naturally extended to kernels over weighted automata. Let A be a weighted automaton defined over the semiring $\mathbb{K}$ and the alphabet Σ and B a weighted automaton defined over the semiring $\mathbb{K}$ and the alphabet Δ, $K(A,B)$ is defined by

$$K(A,B) = \psi\left(\bigoplus_{(x,y) \in \Sigma^* \times \Delta^*} [\![A]\!](x) \otimes [\![T]\!](x,y) \otimes [\![B]\!](y)\right) \tag{1}$$

3. We chose to call these kernels "rational" because their definition is based on *rational relations* or *rational transductions* (Salomaa and Soittola, 1978; Kuich and Salomaa, 1986) represented by a weighted transducer. The mathematical counterpart of weighted automata and transducers are also called *rational power series* Berstel and Reutenauer (1988) which further justifies our terminology.

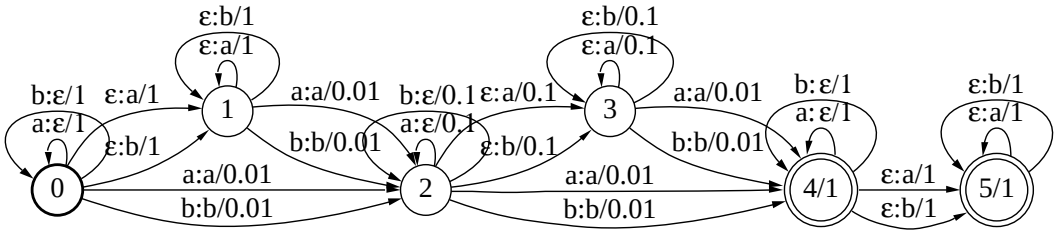


Figure 1: Gappy bigram rational kernel with decay factor $\lambda = .1$. Bold face circles represent initial states and double circles indicate final states. Inside each circle, the first number indicates the state number, the second, at final states only, the value of the final weight function ρ at that state. Arrows represent transitions. They are labeled with an input and an output symbol separated by a colon and followed by their corresponding weight after the slash symbol.

for all weighted automata A and B such that the $\oplus$-sum

$$\bigoplus_{(x,y)\in\Sigma^*\times\Delta^*} [\![A]\!](x)\otimes[\![T]\!](x,y)\otimes[\![B]\!](y)$$

is well-defined and in $\mathbb{K}$. This sum is always defined and in $\mathbb{K}$ when A and B are acyclic weighted automata since the sum then runs over a finite set. It is defined for all weighted automata in all *closed semirings* (Kuich and Salomaa, 1986) such as the tropical semiring. In the probability semiring, the sum is well-defined for all A, B, and T representing probability distributions. When $K(A,B)$ is defined, Equation 1 can be equivalently written as

$$K(A,B) = \psi\left(\bigoplus_{(x,y)\in\Sigma^*\times\Delta^*} [\![A\circ T\circ B]\!](x,y)\right) = \psi(w[A\circ T\circ B]). \tag{2}$$

The next section presents a general algorithm for computing rational kernels.

4. Algorithms

The algorithm for computing $K(x,y)$, or $K(A,B)$, for any two acyclic weighted automata, or for any two weighted automata such that the sum above is well-defined, is based on two general algorithms that we briefly present: composition of weighted transducers to combine A, T, and B, and a general shortest-distance algorithm in a semiring $\mathbb{K}$ to compute the $\oplus$-sum of the weights of all successful paths of the composed transducer.

4.1 Composition of weighted transducers

There exists a general and efficient composition algorithm for weighted transducers which takes advantage of the sparseness of the input transducers (Pereira and Riley, 1997; Mohri et al., 1996). States in the composition $T_1\circ T_2$ of two weighted transducers T_1 and T_2 are identified with pairs of

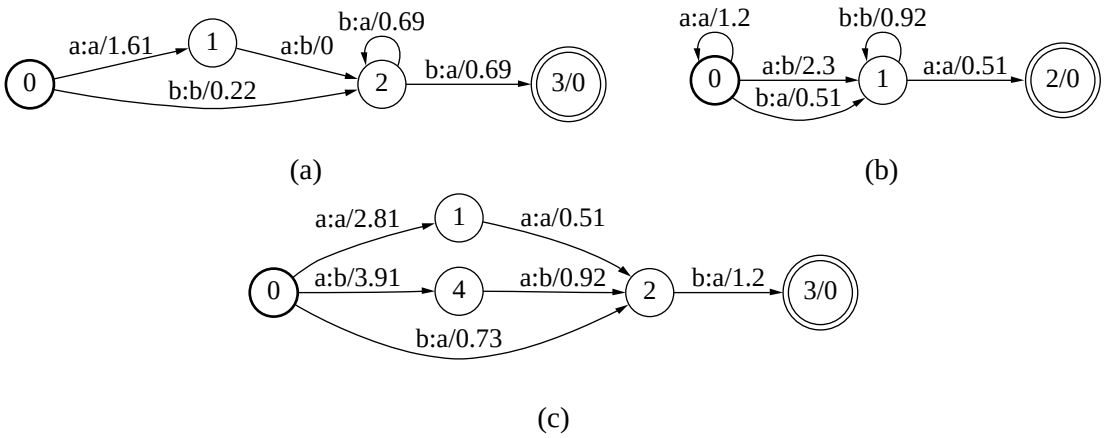


Figure 2: (a) Weighted transducer T_1 over the log semiring. (b) Weighted transducer T_2 over the log semiring. (c) $T_1 \circ T_2$, result of the composition of T_1 and T_2.

a state of T_1 and a state of T_2. Leaving aside transitions with ε inputs or outputs, the following rule specifies how to compute a transition of $T_1 \circ T_2$ from appropriate transitions of T_1 and T_2:[4]

$$(q_1, a, b, w_1, q_2) \quad \text{and} \quad (q_1', b, c, w_2, q_2') \Longrightarrow ((q_1, q_1'), a, c, w_1 \otimes w_2, (q_2, q_2')).$$

In the worst case, all transitions of T_1 leaving a state q_1 match all those of T_2 leaving state q_1', thus the space and time complexity of composition is quadratic: $O((|Q_1| + |E_1|)(|Q_2| + |E_2|))$. Figure 2 illustrates the algorithm when applied to the transducers of Figure 2 (a)-(b) defined over the log semiring.

4.2 Single-source shortest distance algorithm over a semiring

Given a weighted automaton or transducer M, the *shortest-distance* from state q to the set of final states F is defined as the $\oplus$-sum of all the paths from q to F,

$$d[q] = \bigoplus_{\pi \in P(q,F)} w[\pi] \otimes \rho[n[\pi]], \tag{3}$$

when this sum is well-defined and in $\mathbb{K}$. This is always the case when the semiring is *k-closed* or when M is acyclic (Mohri, 2002), the case of interest in our experiments. There exists a general algorithm for computing the shortest-distance $d[q]$ (Mohri, 2002). The algorithm is based on a generalization to k-closed semirings of the relaxation technique used in classical single-source shortest-paths algorithms. When M is acyclic, the complexity of the algorithm is linear: $O(|Q| + (T_\oplus + T_\otimes)|E|)$, where $T_\oplus$ denotes the maximum time to compute $\oplus$ and $T_\otimes$ the time to compute $\otimes$ (Mohri, 2002). The algorithm can then be viewed as a generalization of Lawler's algorithm (Lawler, 1976) to the case of an arbitrary semiring $\mathbb{K}$. It is then based on a generalized relaxation of the outgoing transitions of each state of M visited in reverse topological order (Mohri, 2002).

4. See Pereira and Riley (1997) and Mohri et al. (1996) for a detailed presentation of the algorithm including the use of a transducer filter for dealing with ε-multiplicity in the case of non-idempotent semirings.

Let K be a rational kernel and let T be the associated weighted transducer. Let A and B be two acyclic weighted automata or, more generally, two weighted automata such that the sum in Equation 2 is well-defined and in $\mathbb{K}$. A and B may represent just two strings $x, y \in \Sigma^*$ or may be any complex weighted automata. By definition of rational kernels (Equation 2) and the shortest-distance (Equation 3), $K(A,B)$ can be computed by:

1. Constructing the composed transducer $N = A \circ T \circ B$.
2. Computing $w[N]$, by determining the shortest-distance from the initial states of N to its final states using the shortest-distance algorithm just described.
3. Computing $\psi(w[N])$.

When A and B are acyclic, the shortest-distance algorithm is linear and the total complexity of the algorithm is $O(|T||A||B| + \Phi)$, where $|T|$, $|A|$, and $|B|$ denote respectively the size of T, A and B and Φ the worst case complexity of computing $\psi(x)$, $x \in \mathbb{K}$. If we assume that Φ can be computed in constant time as in many applications, then the complexity of the computation of $K(A,B)$ is quadratic with respect to A and B: $O(|T||A||B|)$.

5. Theory of PDS and NDS Rational Kernels

In learning techniques such as those based on SVMs, we are particularly interested in kernels that are *positive definite symmetric* (PDS), or, equivalently, kernels verifying Mercer's condition, which guarantee the existence of a Hilbert space and a dot product associated to the kernel considered. This ensures the convergence of the training algorithm to a unique optimum. Thus, in what follows, we will focus on theoretical results related to the construction of rational kernels that are PDS. Due to the symmetry condition, the input and output alphabets Σ and Δ will coincide for the underlying transducers associated to the kernels.

This section reviews a number of results related to general PDS kernels, that is the class of all kernels that have the Mercer property (Berg et al., 1984). It also gives novel proofs and results in the specific case of *PDS rational kernels*. These results can be used to combine PDS rational kernels to design new PDS rational kernels or to construct a PDS rational kernel. Our proofs and results are original and are not just straightforward extensions of those existing in the case of general PDS kernels. This is because, for example, a closure property for PDS rational kernels must guarantee not just that the PDS property is preserved but also that the rational property is retained. Our original results include a general construction of PDS rational kernels from arbitrary weighted transducers, a number of theorems related to the converse, and a study of the *negative definiteness* of some rational kernels.

Definition 4 *Let X be a non-empty set. A function $K : X \times X \to \mathbb{R}$ is said to be a PDS kernel if it is symmetric ($K(x,y) = K(y,x)$ for all $x, y \in X$) and*

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

for all $n \geq 1$, $\{x_1, \ldots, x_n\} \subseteq X$ and $\{c_1, \ldots, c_n\} \subseteq \mathbb{R}$.

It is clear from classical results of linear algebra that K is a PDS kernel iff the matrix $K(x_i,x_j)_{i,j\leq n}$ for all $n \geq 1$ and all $\{x_1,\ldots,x_n\} \subseteq X$ is symmetric and all its eigenvalues are non-negative.

PDS kernels can be used to construct other families of kernels that also meet these conditions (Schölkopf and Smola, 2002). *Polynomial kernels* of degree p are formed from the expression $(K+a)^p$, and *Gaussian kernels* can be formed as $\exp(-d^2/\sigma^2)$ with $d^2(x,y) = K(x,x)+K(y,y)-2K(x,y)$. The following sections will provide other ways of constructing PDS rational kernels.

5.1 General Closure Properties of PDS Kernels

The following theorem summarizes general closure properties of PDS kernels (Berg et al., 1984).

Theorem 5 *Let X and Y be two non-empty sets.*

1. Closure under sum*: Let $K_1, K_2 : X \times X \to \mathbb{R}$ be PDS kernels, then $K_1 + K_2 : X \times X \to \mathbb{R}$ is a PDS kernel.*

2. Closure under product*: Let $K_1, K_2 : X \times X \to \mathbb{R}$ be PDS kernels, then $K_1 \cdot K_2 : X \times X \to \mathbb{R}$ is a PDS kernel.*

3. Closure under tensor product*: Let $K_1 : X \times X \to \mathbb{R}$ and $K_2 : Y \times Y \to \mathbb{R}$ be PDS kernels, then their tensor product $K_1 \odot K_2 : (X \times Y) \times (X \times Y) \to \mathbb{R}$, where $K_1 \odot K_2((x_1,y_1),(x_2,y_2)) = K_1(x_1,x_2) \cdot K_2(y_1,y_2)$ is a PDS kernel.*

4. Closure under pointwise limit*: Let $K_n : X \times X \to \mathbb{R}$ be a PDS kernel for all $n \in \mathbb{N}$ and assume that $\lim_{n\to\infty} K_n(x,y)$ exists for all $x,y \in X$, then K defined by $K(x,y) = \lim_{n\to\infty} K_n(x,y)$ is a PDS kernel.*

5. Closure under composition with a power series*: Let $K : X \times X \to \mathbb{R}$ be a PDS kernel such that $|K(x,y)| < \rho$ for all $(x,y) \in X \times X$. Then if the radius of convergence of the power series $S = \sum_{n=0}^{\infty} a_n x^n$ is ρ and $a_n \geq 0$ for all $n \geq 0$, the composed kernel $S \circ K$ is a PDS kernel. In particular, if $K : X \times X \to \mathbb{R}$ is a PDS kernel, then so is $\exp(K)$.*

In particular, these closure properties all apply to PDS kernels that are rational, e.g., the sum or product of two PDS rational kernels is a PDS kernel. However, Theorem 5 does not guarantee the result to be a rational kernel. In the next section, we examine precisely the question of the closure properties of PDS rational kernels (under rational operations).

5.2 Closure Properties of PDS Rational Kernels

In this section, we assume that a fixed function ψ is used in the definition of all the rational kernels mentioned. We denote by K_T the rational kernel corresponding to the transducer T and defined for all $x,y \in \Sigma^*$ by $K_T(x,y) = \psi([\![T]\!](x,y))$.

Theorem 6 *Let Σ be a non-empty alphabet. The following closure properties hold for PDS rational kernels.*

1. Closure under $\oplus$-sum*: Assume that* $\psi : (\mathbb{K}, \oplus, \overline{0}) \to (\mathbb{R}, +, 0)$ *is a monoid morphism.*[5] *Let* $K_{T_1}, K_{T_2} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ *be PDS rational kernels, then* $K_{T_1 \oplus T_2} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ *is a PDS rational kernel and* $K_{T_1 \oplus T_2} = K_{T_1} + K_{T_2}$.

2. Closure under $\otimes$-product*: Assume that* $\psi : (\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1}) \to (\mathbb{R}, +, \times, 0, 1)$ *is a semiring morphism. Let* $K_{T_1}, K_{T_2} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ *be PDS rational kernels, then* $K_{T_1 \otimes T_2} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ *is a PDS rational kernel.*

3. Closure under Kleene-closure*: Assume that* $\psi : (\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1}) \to (\mathbb{R}, +, \times, 0, 1)$ *is a continuous semiring morphism. Let* $K_T : \Sigma^* \times \Sigma^* \to \mathbb{R}$ *be a PDS rational kernel, then* $K_{T^*} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ *is a PDS rational kernel.*

Proof The closure under $\oplus$-sum follows directly from Theorem 5 and the fact that for all $x, y \in \Sigma^*$:

$$\psi([\![T_1]\!](x,y) \oplus [\![T_2]\!](x,y)) = \psi([\![T_1]\!](x,y)) + \psi([\![T_2]\!](x,y))$$

when $\psi : (\mathbb{K}, \oplus, \overline{0}) \to (\mathbb{R}, +, 0)$ is a monoid morphism. For the closure under $\otimes$-product, when ψ is a semiring morphism, for all $x, y \in \Sigma^*$:

$$\begin{aligned} \psi([\![T_1 \otimes T_2]\!](x,y)) &= \sum_{x_1 x_2 = x, y_1 y_2 = y} \psi([\![T_1]\!](x_1,y_1)) \cdot \psi([\![T_2]\!](x_2,y_2)) \\ &= \sum_{x_1 x_2 = x, y_1 y_2 = y} K_{T_1} \odot K_{T_2}((x_1,x_2),(y_1,y_2)). \end{aligned}$$

By Theorem 5, since K_{T_1} and K_{T_2} are PDS kernels, their tensor product $K_{T_1} \odot K_{T_2}$ is a PDS kernel and there exists a Hilbert space $H \subseteq \mathbb{R}^{\Sigma^*}$ and a mapping $u \to \phi_u$ such that $K_{T_1} \odot K_{T_2}(u,v) = \langle \phi_u, \phi_v \rangle$ (Berg et al., 1984). Thus

$$\begin{aligned} \psi([\![T_1 \otimes T_2]\!](x,y)) &= \sum_{x_1 x_2 = x, y_1 y_2 = y} \langle \phi_{(x_1,x_2)}, \phi_{(y_1,y_2)} \rangle \\ &= \left\langle \sum_{x_1 x_2 = x} \phi_{(x_1,x_2)}, \sum_{y_1 y_2 = y} \phi_{(y_1,y_2)} \right\rangle. \end{aligned}$$

Since a dot product is positive definite, this equality implies that $K_{T_1 \otimes T_2}$ is a PDS kernel. A similar proof is given by Haussler (1999). The closure under Kleene-closure is a direct consequence of the closure under $\oplus$-sum and $\otimes$-product of PDS rational kernels and the closure under pointwise limit of PDS kernels (Theorem 5). ■

Theorem 6 provides a general method for constructing complex PDS rational kernels from simpler ones. PDS rational kernels defined to model specific prior knowledge sources can be combined using rational operations to create a more general PDS kernel.

In contrast to Theorem 6, PDS rational kernels are not closed under composition. This is clear since the ordinary matrix multiplication does not preserve positive definiteness in general.

The next section studies a general construction of PDS rational kernels using composition.

5. A *monoid morphism* $\psi : (\mathbb{K}, \oplus, \overline{0}) \to (\mathbb{R}, +, 0)$ is a function verifying $\psi(x \oplus y) = \psi(x) + \psi(y)$ for all $x, y \in \mathbb{K}$, and $\psi(\overline{0}) = 0$. A *semiring morphism* ψ is a function $\psi : (\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1}) \to (\mathbb{R}, +, \times, 0, 1)$ further verifying $\psi(x \otimes y) = \psi(x) \cdot \psi(y)$ for all $x, y \in \mathbb{K}$, and $\psi(\overline{1}) = 1$.

5.3 A General Construction of PDS Rational Kernels

In this section, we assume that $\psi : (\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1}) \to (\mathbb{R}, +, \times, 0, 1)$ is a continuous semiring morphism. This limits the choice of the semiring associated to the weighted transducer defining a rational kernel, since it needs in particular to be commutative and non-idempotent.[6] Our study of PDS rational kernels in this section is thereby limited to such semirings. This should not leave the reader with the incorrect perception that all PDS rational kernels are defined over non-idempotent semirings though. As already mentioned before, in general, the function ψ can be chosen arbitrarily and needs not impose any algebraic property on the semiring used.

We show that there exists a general way of constructing a PDS rational kernel from any weighted transducer T. The construction is based on composing T with its inverse T^{-1}.

Proposition 7 *Let $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ be a weighted finite-state transducer defined over the semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$. Assume that the weighted transducer $T \circ T^{-1}$ is regulated, then $(\psi, T \circ T^{-1})$ defines a PDS rational kernel over $\Sigma^* \times \Sigma^*$.*

Proof Denote by S the composed transducer $T \circ T^{-1}$. Let K be the rational kernel defined by S. By definition of composition,

$$K(x,y) = \psi([\![S]\!](x,y)) = \psi\left(\bigoplus_{z \in \Delta^*} [\![T]\!](x,z) \otimes [\![T]\!](y,z)\right),$$

for all $x, y \in \Sigma^*$. Since ψ is a continuous semiring morphism, for all $x, y \in \Sigma^*$,

$$K(x,y) = \psi([\![S]\!](x,y)) = \sum_{z \in \Delta^*} \psi([\![T]\!](x,z)) \cdot \psi([\![T]\!](y,z)).$$

For all $n \in \mathbb{N}$ and $x, y \in \Sigma^*$, define $K_n(x,y)$ by

$$K_n(x,y) = \sum_{|z| \leq n} \psi([\![T]\!](x,z)) \cdot \psi([\![T]\!](y,z)),$$

where the sum runs over all strings $z \in \Delta^*$ of length less than or equal to n. Clearly, K_n defines a symmetric kernel. For any $l \geq 1$ and any $x_1, \ldots, x_l \in \Sigma^*$, define the matrix M_n by $M_n = (K_n(x_i, x_j))_{i \leq l, j \leq l}$. Let $z_1, z_2, \ldots, z_m$ be an arbitrary ordering of the strings of length less than or equal to n. Define the matrix A by

$$A = (\psi([\![T]\!](x_i, z_j)))_{i \leq l; j \leq m}.$$

By definition of K_n, $M_n = AA^t$. The eigenvalues of AA^t are non-negative for any rectangular matrix A, thus K_n is a PDS kernel. Since K is a pointwise limit of K_n, $K(x,y) = \lim_{n \to \infty} K_n(x,y)$, by Theorem 5, K is a PDS kernel. This ends the proof of the proposition. ■

The next propositions provide results related to the converse of Proposition 7. We denote by $Id_{\mathbb{R}}$ the identity function over $\mathbb{R}$.

Proposition 8 *Let $S = (\Sigma, \Sigma, Q, I, F, E, \lambda, \rho)$ be an acyclic weighted finite-state transducer over $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ such that (ψ, S) defines a PDS rational kernel on $\Sigma^* \times \Sigma^*$, then there exists a weighted transducer T over the probability semiring such that $(Id_{\mathbb{R}}, T \circ T^{-1})$ defines the same rational kernel.*

6. If $\mathbb{K}$ is idempotent, for any $x \in \mathbb{K}$, $\psi(x) = \psi(x \oplus x) = \psi(x) + \psi(x) = 2\psi(x)$, which implies that $\psi(x) = 0$ for all x.

Proof Let S be an acyclic weighted transducer verifying the hypotheses of the proposition. Let $X \subset \Sigma^*$ be the finite set of strings accepted by S. Since S is symmetric, $X \times X$ is the set of pairs of strings (x,y) defining the rational relation associated with S. Let $x_1, x_2, \ldots, x_n$ be an arbitrary numbering of the elements of X. Define the matrix M by

$$M = (\psi([\![S]\!](x_i, x_j)))_{1 \leq i \leq n, 1 \leq j \leq n}.$$

Since S defines a PDS rational kernel, M is a symmetric matrix with non-negative eigenvalues, i.e., M is symmetric positive semi-definite. The Cholesky decomposition extends to the case of semi-definite matrices (Dongarra et al., 1979): there exists an upper triangular matrix $R = (R_{ij})$ with non-negative diagonal elements such that $M = RR^t$. Let $Y = \{y_1, \ldots, y_n\}$ be an arbitrary subset of n distinct strings of Σ^*. Define the weighted transducer T over the $X \times Y$ by

$$[\![T]\!](x_i, y_j) = R_{ij}$$

for all i, j, $1 \leq i, j \leq n$. By definition of composition, $[\![T \circ T^{-1}]\!](x_i, x_j) = \psi([\![S]\!](x_i, x_j))$ for all i, j, $1 \leq i, j \leq n$. Thus, $T \circ T^{-1} = \psi(S)$, which proves the claim of the proposition. ■

Note that when the matrix M introduced in the proof is positive definite, that is when the eigenvalues of the matrix associated with S are all positive, then Cholesky's decomposition and thus the weights associated to the input strings of T are unique.

Assume that the same continuous semiring morphism ψ is used in the definition of all the rational kernels.

Proposition 9 *Let Θ be the subset of the weighted transducers over $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ such that for any $S \in \Theta$, (ψ, S) defines a PDS rational kernel and there exists a weighted transducer $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ over the probability semiring such that $(Id_{\mathbb{R}}, T \circ T^{-1})$ defines the same rational kernel as (ψ, S). Then Θ is closed under $\oplus$-sum, $\otimes$-product, and Kleene-closure.*

Proof Let $S_1, S_2 \in \Theta$, we will show that $S_1 \oplus S_2 \in \Theta$, $S_1 \otimes S_2 \in \Theta$, and $S_1^* \in \Theta$. By definition, there exist $T_1 = (\Sigma, \Delta_1, Q_1, I_1, F_1, E_1, \lambda_1, \rho_1)$ and $T_2 = (\Sigma, \Delta_2, Q_2, I_2, F_2, E_2, \lambda_2, \rho_2)$ such that

$$K_1 = T_1 \circ T_1^{-1} \quad \text{and} \quad K_2 = T_2 \circ T_2^{-1},$$

where K_1 (K_2) is the PDS rational kernel defined by (ψ, S_1) (resp. (ψ, S_2)). Let u be an alphabetic morphism mapping Δ_2 to a new alphabet Δ_2' such that $\Delta_1 \cap \Delta_2' = \emptyset$. u is clearly a rational transduction (Berstel, 1979) and can be represented by a finite-state transducer U. Thus, we can define a new weighted transducer T_2' by $T_2' = T_2 \circ U = (\Sigma, \Delta_2', Q_2, I_2, F_2, E_2', \lambda_2, \rho_2)$, which only differs from T_2 by some renaming of its output labels. This does not affect the result of the composition with the inverse transducer since $U \circ U^{-1}$ is the identity mapping over Δ_2^*:

$$T_2' \circ T'^{-1}_2 = T_2 \circ U \circ (U^{-1} \circ T_2^{-1}) = T_2 \circ T_2^{-1} = K_2. \tag{4}$$

Since, T_1 and T_2 have distinct output alphabets, their output labels cannot match; thus

$$T_1 \circ T'^{-1}_2 = \emptyset \quad \text{and} \quad T_2' \circ T_1^{-1} = \emptyset. \tag{5}$$

Let $T = T_1 + T_2'$, in view of Equation 4 and Equation 5:

$$T \circ T^{-1} = (T_1 + T_2') \circ (T_1 + T_2')^{-1} = (T_1 \circ T_1^{-1}) + (T'_2 \circ T'^{-1}_2) = K_1 + K_2.$$

Since the same continuous semiring morphism ψ is used for the definition of all the rational kernels in Θ, by Theorem 6, $K_1 + K_2$ is a PDS rational kernel defined by $(\psi, S_1 \oplus S_2)$ and $S_1 \oplus S_2$ is in Θ. Similarly, define T' as $T' = T_1 \cdot T_2'$:

$$T' \circ T'^{-1} = (T_1 \cdot T_2') \circ (T_1 \cdot T_2')^{-1} = (T_1 \circ T_1^{-1}) \cdot (T_2 \circ T'^{-1}_2).$$

Thus, $S_1 \otimes S_2$ is in Θ. Let x be a symbol not in Δ_1 and let $\Delta_1' = \Delta_1 \cup \{x\}$. Let V be the finite-state transducer accepting as input only ε and mapping ε to x and define T_1' by $T_1' = V \cdot T_1$. Since x does not match any of the output labels of T_1, $T_1' \circ T_1'^{-1} = T_1 \circ T_1{}^{-1}$ and $(T_1' \circ T_1'^{-1})^* = T_1'^* \circ (T_1'^{-1})^*$:

$$(T_1 \circ T_1{}^{-1})^* = (T_1' \circ T_1'^{-1})^* = T_1'^* \circ (T_1'^{-1})^*.$$

Thus, by Theorem 6, S_1^* is a PDS rational kernel that is in Θ. ■

Proposition 9 raises the following question: under the same assumptions, are all PDS rational kernels defined by a pair of the form $(\psi, T \circ T^{-1})$? A natural conjecture is that this is the case and that this property provides a characterization of the weighted transducers defining PDS rational kernels. Propositions 8 and 9 both favor that conjecture. Proposition 8 shows that this holds in the acyclic case. Proposition 9 might be useful to extend this to the general case.

In the case of PDS rational kernels defined by $(Id_{\mathbb{R}}, S)$ with S a weighted transducer over the probability semiring, the conjecture could be reformulated as: is S of the form $S = T \circ T^{-1}$? If true, this could be viewed as a generalization of Cholesky's decomposition theorem to the case of infinite matrices given by weighted transducers over the probability semiring.

This ends our discussion of PDS rational kernels. In the next section, we will examine *negative definite kernels* and their relationship with PDS rational kernels.

5.4 Negative Definite Kernels

As mentioned before, given a set X and a distance or dissimilarity measure $d : X \times X \to \mathbb{R}_+$, a common method used to define a kernel K is the following. For all $x, y \in X$,

$$K(x,y) = \exp(-td^2(x,y)),$$

where $t > 0$ is some constant typically used for normalization. Gaussian kernels are defined in this way. However, such kernels K are not necessarily positive definite, e.g., for $X = \mathbb{R}$, $d(x,y) = |x-y|^p$, $p > 1$ and $t = 1$, K is not positive definite. The positive definiteness of K depends on t and the properties of the function d. The classical results presented in this section exactly address such questions (Berg et al., 1984). They include a characterization of PDS kernels based on *negative definite kernels* which may be viewed as distances with some specific properties.[7]

The results we are presenting are general, but we are particularly interested in the case where d can be represented by a rational kernel. We will use these results later when dealing with the case of the edit-distance.

7. Many of the results given by Berg et al. (1984) are re-presented in (Schölkopf, 2001) with the terminology of *conditionally positive definite* instead of *negative definite kernels*. We adopt the original terminology used by Berg et al. (1984).

Definition 10 *Let X be a non-empty set. A function $K : X \times X \to \mathbb{R}$ is said to be a* negative definite symmetric kernel *(NDS kernel) if it is symmetric ($K(x,y) = K(y,x)$ for all $x,y \in X$) and*

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \leq 0$$

for all $n \geq 1$, $\{x_1, \ldots, x_n\} \subseteq X$ and $\{c_1, \ldots, c_n\} \subseteq \mathbb{R}$ with $\sum_{i=1}^{n} c_i = 0$.

Clearly, if K is a PDS kernel then $-K$ is a NDS kernel, however the converse does not hold in general. Negative definite kernels often correspond to distances, e.g., $K(x,y) = (x-y)^{\alpha}$, $x,y \in \mathbb{R}$, with $0 < \alpha \leq 2$ is a negative definite kernel.

The next theorem summarizes general closure properties of NDS kernels (Berg et al., 1984).

Theorem 11 *Let X be a non-empty set.*

1. Closure under sum*: Let $K_1, K_2 : X \times X \to \mathbb{R}$ be NDS kernels, then $K_1 + K_2 : X \times X \to \mathbb{R}$ is a NDS kernel.*

2. Closure under log and exponentiation*: Let $K : X \times X \to \mathbb{R}$ be a NDS kernel with $K \geq 0$, and α a real number with $0 < \alpha < 1$, then $\log(1+K), K^{\alpha} : X \times X \to \mathbb{R}$ are NDS kernels.*

3. Closure under pointwise limit*: Let $K_n : X \times X \to \mathbb{R}$ be a NDS kernel for all $n \in \mathbb{N}$, then K defined by $K(x,y) = \lim_{n \to \infty} K_n(x,y)$ is a NDS kernel.*

The following theorem clarifies the relation between NDS and PDS kernels and provides in particular a way of constructing PDS kernels from NDS ones (Berg et al., 1984).

Theorem 12 *Let X be a non-empty set, $x_o \in X$, and let $K : X \times X \to \mathbb{R}$ be a symmetric kernel.*

1. *K is negative definite iff $\exp(-tK)$ is positive definite for all $t > 0$.*

2. *Let K' be the function defined by*

$$K'(x,y) = K(x,x_0) + K(y,x_0) - K(x,y) - K(x_0,x_0).$$

Then K is negative definite iff K' is positive definite.

The theorem gives two ways of constructing a positive definite kernel using a negative definite kernel. The first construction is similar to the way Gaussian kernels are defined. The second construction has been put forward by (Schölkopf, 2001).

6. Relationship with some commonly used kernels or similarity measures

This section studies the relationships between several families of kernels or similarities measures and rational kernels.

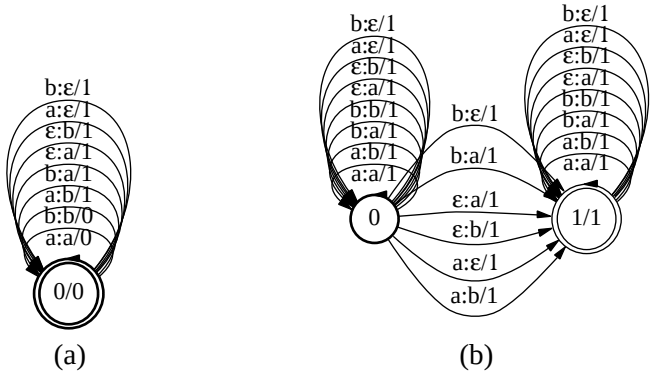


Figure 3: (a) Weighted transducer over the tropical semiring representing the edit-distance over the alphabet $\Sigma = \{a, b\}$. (b) Weighted transducer over the probability semiring computing the cost of alignments over the alphabet $\Sigma = \{a, b\}$.

6.1 Edit-Distance

A common similarity measure in many applications is that of the *edit-distance*, that is the minimal cost of a series of edit operations (symbol insertions, deletions, or substitutions) transforming one string into the other (Levenshtein, 1966). We denote by $d_e(x,y)$ the edit-distance between two strings x and y over the alphabet Σ with cost 1 assigned to all edit operations.

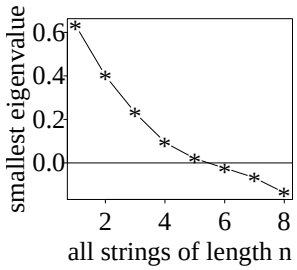
Proposition 13 *Let Σ be a non-empty finite alphabet and let d_e be the edit-distance over Σ, then d_e is a symmetric rational kernel. Furthermore, (1): d_e is not a PDS kernel, and (2): d_e is a NDS kernel iff $|\Sigma| = 1$.*

Proof The edit-distance between two strings, or weighted automata, can be represented by a simple weighted transducer over the tropical semiring (Mohri, 2003). Since the edit-distance is symmetric, d_e is a symmetric rational kernel. Figure 3(a) shows the corresponding transducer when the alphabet is $\Sigma = \{a, b\}$. The cost of the alignment between two sequences can also be computed by a weighted transducer over the probability semiring (Mohri, 2003), see Figure 3(b).

Let $a \in \Sigma$, then the matrix $(d_e(x_i, x_j))_{1 \leq i,j \leq 2}$ with $x_1 = \varepsilon$ and $x_2 = a$ has a negative eigenvalue (-1), thus d_e is not a PDS kernel.

When $|\Sigma| = 1$, the edit-distance simply measures the absolute value of the difference of length between two strings. A string $x \in \Sigma^*$ can then be viewed as a vector of the Hilbert space $\mathbb{R}^\infty$. Denote by $\|\cdot\|$ the corresponding norm. For all $x, y \in \Sigma^*$,

$$d_e(x,y) = \|x - y\|.$$



(a)

i	1	2	3	4	5
x_i	abc	bad	dab	adc	bcd
c_i	1	1	$-\frac{2}{3}$	$-\frac{2}{3}$	$-\frac{2}{3}$

(b)

Figure 4: (a) Smallest eigenvalue of the matrix $M_n = (\exp(-d_e(x_i,x_j)))_{1\leq i,j,\leq 2^n}$ as a function of n. (b) Example demonstrating that the edit-distance is not negative definite.

The square distance $\|\cdot\|^2$ is negative definite, thus by Theorem 11, $d_e = (\|\cdot\|^2)^{1/2}$ is also negative definite.

Assume now that $|\Sigma| > 1$. We show that $\exp(-d_e)$ is not PDS. By Theorem 12, this implies that d_e is not negative definite. Let $x_1, \cdots, x_{2^n}$ be any ordering of the strings of length n over the alphabet $\{a,b\}$. Define the matrix M_n by

$$M_n = (\exp(-d_e(x_i,x_j)))_{1\leq i,j,\leq 2^n}.$$

Figure 4(a) shows the smallest eigenvalue α_n of M_n as a function of n. Clearly, there are values of n for which $\alpha_n < 0$, thus the edit-distance is not negative definite. Table 4(b) provides a simple example with five strings of length 3 over the alphabet $\Sigma = \{a,b,c,d\}$ showing directly that the edit-distance is not negative definite. Indeed, it is easy to verify that $\sum_{i=1}^5 \sum_{j=1}^5 c_i c_j K(x_i,x_j) = \frac{2}{3} > 0$. ■

To our knowledge, this is the first statement and proof of the fact that d_e is not NDS for $|\Sigma| > 1$. This result has a direct consequence on the design of kernels in computational biology, often based on the edit-distance or other related similarity measures. The edit-distance and other related similarity measures are often used in computational biology. When $|\Sigma| > 1$, Proposition 13 shows that d_e is not NDS. Thus, there exists $t > 0$ for which $\exp(-td_e)$ is not PDS. Similarly, d_e^2 is not NDS since otherwise by Theorem 11, $d_e = (d_e^2)^{1/2}$ would be NDS.

6.2 Haussler's Convolution Kernels for Strings

D. Haussler describes a class of kernels for strings built by applying iteratively *convolution kernels* (Haussler, 1999). We show that these convolution kernels for strings are specific instances of rational kernels. Haussler (1999) defines the *convolution* of two string kernels K_1 and K_2 over the alphabet Σ as the kernel denoted by $K_1 \star K_2$ and defined for all $x,y \in \Sigma^*$ by

$$K_1 \star K_2(x,y) = \sum_{x_1x_2=x, y_1y_2=y} K_1(x_1,y_1) \cdot K_2(x_2,y_2).$$

Clearly, when K_1 and K_2 are given by weighted transducers over the probability semiring, this definition coincides with that of the product (or concatenation) of transducers (Equation 1). Haussler

(1999) also introduces for $0 \leq \gamma < 1$ the γ-infinite iteration of a mapping $H : \Sigma^* \times \Sigma^* \to \mathbb{R}$ by

$$H^*_\gamma = (1-\gamma) \sum_{n=1}^{\infty} \gamma^{n-1} H^{(n)},$$

where $H^{(n)} = H \star H^{(n-1)}$ is the result of the convolution of H with itself $n-1$ times. Note that $H^*_\gamma = 0$ for $\gamma = 0$.

Lemma 14 *For $0 < \gamma < 1$, the γ-infinite iteration of a rational transduction $H : \Sigma^* \times \Sigma^* \to \mathbb{R}$ can be defined in the following way with respect to the Kleene $\dagger$-operator:*

$$H^*_\gamma = \frac{1-\gamma}{\gamma} (\gamma H)^\dagger.$$

Proof Haussler's convolution simply corresponds to the product (or concatenation) in the case of rational transductions. Thus, for $0 < \gamma < 1$, by definition of the $\dagger$-operator,

$$(\gamma H)^\dagger = \sum_{n=1}^{\infty} (\gamma H)^n = \sum_{n=1}^{\infty} \gamma^n H^n = \frac{\gamma}{1-\gamma} \sum_{n=1}^{\infty} (1-\gamma)\gamma^{n-1} H^n = \frac{\gamma}{1-\gamma} H^*_\gamma \qquad .$$

■

Given a probability distribution p over all symbols of Σ, Haussler's convolution kernels for strings are defined by

$$K_H(x,y) = \gamma K_2 \star (K_1 \star K_2)^\star_\gamma + (1-\gamma) K_2,$$

where K_1 is the specific polynomial PDS rational transduction over the probability semiring defined by $K_1(x,y) = \sum_{a \in \Sigma} p(x|a)p(y|a)p(a)$ and models substitutions, and K_2 another specific PDS rational transduction over the probability semiring modeling insertions.

Proposition 15 *For any $0 \leq \gamma < 1$, Haussler's convolution kernels K_H coincide with the following special cases of rational kernels:*

$$K_H = (1-\gamma)[K_2(\gamma K_1 K_2)^*].$$

Proof As mentioned above, Haussler's convolution simply corresponds to concatenation in this context. When $\gamma = 0$, by definition, K_H is reduced to K_2 which is a rational transducer and the proposition's formula above is satisfied. Assume now that $\gamma \neq 0$. By lemma 14, K_H can be re-written as

$$\begin{aligned} K_H &= \gamma K_2 (K_1 K_2)^\star_\gamma + (1-\gamma) K_2 = \gamma K_2 \frac{1-\gamma}{\gamma} (\gamma K_1 K_2)^\dagger + (1-\gamma) K_2 \\ &= (1-\gamma)[K_2(\gamma K_1 K_2)^\dagger + K_2] = (1-\gamma)[K_2(\gamma K_1 K_2)^*]. \end{aligned}$$

Since rational transductions are closed under rational operations, K_H also defines a rational transduction. Since K_1 and K_2 are PDS kernels, by Theorem 6, K_H defines a PDS kernel. ■

The transducer of Figure 5 illustrates the convolution kernels for strings proposed by Haussler. They correspond to special cases of rational kernels whose mechanism is clarified by the figure: the kernel corresponds to an insertion with weight $(1-\gamma)$ modeled by K_2 followed by any number of sequences of substitutions modeled by K_1 and insertions modeled by K_2 with weight γ. Clearly, there are many other ways of defining kernels based on weighted transducers with more complex definitions and perhaps more data-driven definitions.

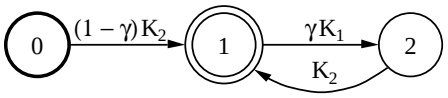


Figure 5: Haussler's convolution kernels K_H for strings: specific instances of rational kernels. K_1, (K_2), corresponds to a specific weighted transducer over the probability semiring and modeling substitutions (resp. insertions).

6.3 Other Kernels Used in Computational Biology

In this section we show the relationship between rational kernels and another class of kernels used in computational biology.

A family of kernels, *mismatch string kernels*, was introduced by (Leslie et al., 2003) for protein classification using SVMs. Let Σ be a finite alphabet, typically that of amino acids for protein sequences. For any two sequences $z_1, z_2 \in \Sigma^*$ of same length ($|z_1| = |z_2|$), we denote by $d(z_1, z_2)$ the total number of mismatching symbols between these sequences. For all $m \in \mathbb{N}$, we define the bounded distance d_m between two sequences of same length by

$$d_m(z_1, z_2) = \begin{cases} 1 & \text{if } (d(z_1, z_2) \leq m) \\ 0 & \text{otherwise.} \end{cases}$$

and for all $k \in \mathbb{N}$, we denote by $F_k(x)$ the set of all factors of x of length k:

$$F_k(x) = \{z : x \in \Sigma^* z \Sigma^*, |z| = k\}.$$

For any $k, m \in \mathbb{N}$ with $m \leq k$, a (k,m)-mismatch kernel $K_{(k,m)} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is the kernel defined over protein sequences $x, y \in \Sigma^*$ by

$$K_{(k,m)}(x,y) = \sum_{z_1 \in F_k(x), z_2 \in F_k(y), z \in \Sigma^k} d_m(z_1, z)\, d_m(z, z_2).$$

Proposition 16 *For any $k, m \in \mathbb{N}$ with $m \leq k$, the (k,m)-mismatch kernel $K_{(k,m)} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is a PDS rational kernel.*

Proof Let M, S, and D be the weighted transducers over the probability semiring defined by

$$M = \sum_{a \in \Sigma} (a, a) \qquad S = \sum_{a \neq b} (a, b) \qquad D = \sum_{a \in \Sigma} (a, \varepsilon).$$

M associates weight 1 to each pair of identical symbols of the alphabet Σ, S associates 1 to each pair of distinct or mismatching symbols, and D associates 1 to all pairs with second element ε.

For $i, k \in \mathbb{N}$ with $0 \leq i \leq k$, Define the *shuffle* of S^i and M^{k-i}, denoted by $S^i \sqcup\!\sqcup M^{k-i}$, as the the sum over all products made of factors S and M with exactly i factors S and $k-i$ factors M. As a finite sum of products of S and M, $S^i \sqcup\!\sqcup M^{k-i}$ is rational. Since weighted transducers are closed under rational operations the following defines a weighted transducer T over the probability semiring for

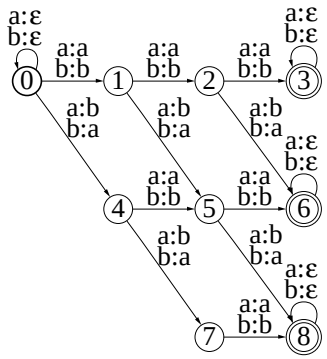


Figure 6: Mismatch kernel $K_{(k,m)} = T_{k,m} \circ T_{k,m}^{-1}$ (Leslie et al., 2003) with $k = 3$ and $m = 2$ and with $\Sigma = \{a, b\}$. The transducer $T_{3,2}$ defined over the probability semiring is shown. All transition weights and final weights are equal to one. Note that states 3, 6, and 8 of the transducer are equivalent and thus can be merged and similarly that states 2 and 5 can then be merged as well.

any $k, m \in \mathbb{N}$ with $m \leq k$: $T_{k,m} = D^* R D^*$ with $R = \sum_{i=0}^{m} S^i \sqcup\!\sqcup M^{k-i}$. Consider two sequences z_1, z_2 such that $|z_1| = |z_2| = k$. By definition of M and S and the shuffle product, for any i, with $0 \leq i \leq m$,

$$[\![S^i \sqcup\!\sqcup M^{k-i}]\!](z_1, z_2) = \begin{cases} 1 & \text{if } (d(z_1, z_2) = i) \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{Thus,} \quad [\![R]\!](z_1, z_2) = \sum_{i=0}^{m} S^i \sqcup\!\sqcup M^{k-i}(z_1, z_2) = \begin{cases} 1 & \text{if } (d(z_1, z_2) \leq m) \\ 0 & \text{otherwise} \end{cases}$$
$$= d_m(z_1, z_2).$$

By definition of the product of weighted transducers, for any $x \in \Sigma^*$ and $z \in \Sigma^k$,

$$T_{k,m}(x, z) = \sum_{x = uvw, z = u'v'w'} [\![D^*]\!](u, u') \, [\![R]\!](v, v') \, [\![D^*]\!](w, w')$$
$$= \sum_{v \in F_k(x), z = v'} [\![R]\!](v, v') = \sum_{v \in F_k(x)} d_m(v, z).$$

It is clear from the definition of $T_{k,m}$ that $T_{k,m}(x, z) = 0$ for all $x, z \in \Sigma^*$ with $|z| > k$. Thus, by definition of the composition of weighted transducer, for all $x, y \in \Sigma^*$

$$[\![T_{k,m} \circ {T_{k,m}}^{-1}]\!](x, y) = \sum_{z_1 \in F_k(x), z_2 \in F_k(y), z \in \Sigma^*} d_m(z_1, z) \, d_m(z, z_2)$$
$$= \sum_{z_1 \in F_k(x), z_2 \in F_k(y), z \in \Sigma^k} d_m(z_1, z) \, d_m(z, z_2) = K_{(k,m)}(x, y).$$

By proposition 7, this proves that $K_{(k,m)}$ is a PDS rational kernel. ■

Figure 6 shows $T_{3,2}$, a simple weighted transducer over the probability semiring that can be used to compute the mismatch kernel $K_{(3,2)} = T_{3,2} \circ {T_{3,2}}^{-1}$. Such transducers provide a compact representation of the kernel and are very efficient to use with the composition algorithm already described in (Cortes et al., 2003c). The transitions of these transducers can be defined implicitly and expanded on-demand as needed for the particular input strings or weighted automata. This substantially reduces the space needed for their representation, e.g., a single transition with labels $x : y$, $x \neq y$ can be used to represent all transitions with similar labels ($(a : b)$, $a, b \in \Sigma$, with $a \neq b$). Similarly, composition can also be performed on-the-fly. Furthermore, the transducer of Figure 6 can be made more compact since it admits several states that are equivalent.

7. Applications and Experiments

Rational kernels can be used in a variety of applications ranging from computational biology to optical character recognition. We have applied them successfully to a number of speech processing tasks including the identification from speech of traits, or *voice signatures*, such as emotion (Shafran et al., 2003). This section describes some of our most recent applications to spoken-dialog classification.

We first introduce a general family of PDS rational kernels relevant to spoken-dialog classification tasks that we used in our experiments, then discuss the spoken-dialog classification problem and report our experimental results.

7.1 A General Family of PDS Kernels: n-gram Kernels

A rational kernel can be viewed as a similarity measure between two sequences or weighted automata. One may for example consider two utterances to be similar when they share many common n-gram subsequences. The exact transcriptions of the utterances are not available but we can use the *word lattices* output by the recognizer instead.

A word lattice is a weighted automaton over the log semiring that compactly represents the most likely transcriptions of a speech utterance. Each path of the automaton is labeled with a sequence of words whose weight is obtained by adding the weights of the constituent transitions. The weight assigned by the lattice to a sequence of words can often be interpreted as the log-likelihood of that transcription based on the models used by the recognizer. More generally, the weights are used to rank possible transcriptions, the sequence with the lowest weight being the most favored transcription.

A word lattice A can be viewed as a probability distribution P_A over all strings $s \in \Sigma^*$. Modulo a normalization constant, the weight assigned by A to a string x is $[\![A]\!](x) = -\log P_A(x)$. Denote by $|s|_x$ the number of occurrences of a sequence x in the string s. The expected count or number of occurrences of an n-gram sequence x in s for the probability distribution P_A is

$$c(A,x) = \sum_s P_A(s)|s|_x.$$

Two lattices output by a speech recognizer can be viewed as similar when the sum of the product of the expected counts they assign to their common n-gram sequences is sufficiently high. Thus, we define an n-gram kernel k_n for two lattices A_1 and A_2 by

$$k_n(A_1,A_2) = \sum_{|x|=n} c(A_1,x)\, c(A_2,x).$$

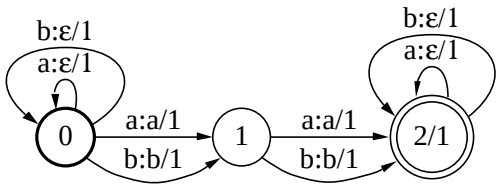


Figure 7: Weighted transducer T computing expected counts of bigram sequences of a word lattice with $\Sigma = \{a, b\}$.

The kernel k_n is a PDS rational kernel of type $T \circ T^{-1}$ and it can be computed efficiently.

Indeed, there exists a simple weighted transducer T that can be used to computed $c(A_1, x)$ for all n-gram sequences $x \in \Sigma^*$. Figure 7 shows that transducer in the case of bigram sequences ($n = 2$) and for the alphabet $\Sigma = \{a, b\}$. The general definition of T is

$$T = (\Sigma \times \{\varepsilon\})^* \, (\sum_{x \in \Sigma} \{x\} \times \{x\})^n \, (\Sigma \times \{\varepsilon\})^*.$$

k_n can be written in terms of the weighted transducer T as

$$\begin{aligned} k_n(A_1, A_2) &= w[(A_1 \circ T) \circ (T^{-1} \circ A_2)] \\ &= w[(A_1 \circ (T \circ T^{-1}) \circ A_2)], \end{aligned}$$

which shows that it is a rational kernel whose associated weighted transducer is $T \circ T^{-1}$. In view of Proposition 7, k_n is a PDS rational kernel. Furthermore, the general composition algorithm and shortest-distance algorithm described in Section 4 can be used to compute k_n efficiently. The size of the transducer T is $O(n|\Sigma|)$ but in practice, a lazy implementation can be used to simulate the presence of the transitions of T labeled with all elements of Σ. This reduces the size of the machine used to $O(n)$. Thus, since the complexity of composition is quadratic (Mohri et al., 1996; Pereira and Riley, 1997) and since the general shortest distance algorithm just mentioned is linear for acyclic graphs such as the lattices output by speech recognizers (Mohri, 2002), the worst case complexity of the algorithm is $O(n^2 \, |A_1| \, |A_2|)$.

By Theorem 6, the sum of two kernels k_n and k_m is also a PDS rational kernel. We define an *n-gram rational kernel* K_n as the PDS rational kernel obtained by taking the sum of all k_m, with $1 \leq m \leq n$:

$$K_n = \sum_{m=1}^{n} k_m.$$

Thus, the feature space associated with K_n is the set of all m-gram sequences with $m \leq n$. n-gram kernels are used in our experiments in spoken-dialog classification.

7.2 Spoken-Dialog Classification

7.2.1 DEFINITION

One of the key tasks of spoken-dialog systems is classification. This consists of assigning, out of a finite set, a specific category to each speech utterance based on the transcription of that utterance by

Dataset	Number of classes	Training size	Testing size	Number of *n*-grams	ASR word accuracy
HMIHY 0300	64	35551	5000	24177	72.5%
VoiceTone1	97	29561	5537	22007	70.5%
VoiceTone2	82	9093	5172	8689	68.8%

Table 2: Key characteristics of the three datasets used in the experiments. The fifth column displays the total number of unigrams, bigrams, and trigrams found in the one-best output of the ASR for the utterances of the training set, that is the number of features used by BoosTexter or SVMs used with the one-best outputs.

a speech recognizer. The choice of possible categories depends on the dialog context considered. A category may correspond to the type of billing problem in the context of a dialog related to billing, or to the type of problem raised by the speaker in the context of a hot-line service. Categories are used to direct the dialog manager in formulating a response to the speaker's utterance. Classification is typically based on features such as relevant key words or key sequences used by a machine learning algorithm.

The word error rate of conversational speech recognition systems is still too high in many tasks to rely only on the one-best output of the recognizer (the word error rate in the deployed services we have experimented with is about 70%, as we will see later). However, the *word lattices* output by speech recognition systems may contain the correct transcription in most cases. Thus, it is preferable to use instead the full word lattices for classification.

The application of classification algorithms to word lattices raises several issues. Even small word lattices may contain billions of paths, thus the algorithms cannot be generalized by simply applying them to each path of the lattice. Additionally, the paths are weighted and these weights must be used to guide appropriately the classification task. The use of rational kernels solves both of these problems since they define kernels between weighted automata and since they can be computed efficiently (Section 4).

7.2.2 DESCRIPTION OF TASKS AND DATASETS

We did a series of experiments in several large-vocabulary spoken-dialog tasks using rational kernels with a twofold objective: to improve classification accuracy in those tasks, and to evaluate the impact on classification accuracy of the use a word lattice rather than the one-best output of the automatic speech recognition (ASR) system.

The first task we considered is that of a deployed customer-care application (HMIHY 0300). In this task, users interact with a spoken-dialog system via the telephone, speaking naturally, to ask about their bills, their calling plans, or other similar topics. Their responses to the open-ended prompts of the system are not constrained by the system, they may be any natural language sequence. The objective of the spoken-dialog classification is to assign one or several categories or call-types, e.g., *Billing Credit*, or *Calling Plans*, to the users' speech utterances. The set of categories is finite and is limited to 64 classes. The calls are classified based on the user's response to the first greeting prompt: "*Hello, this is AT&T. How may I help you?*".

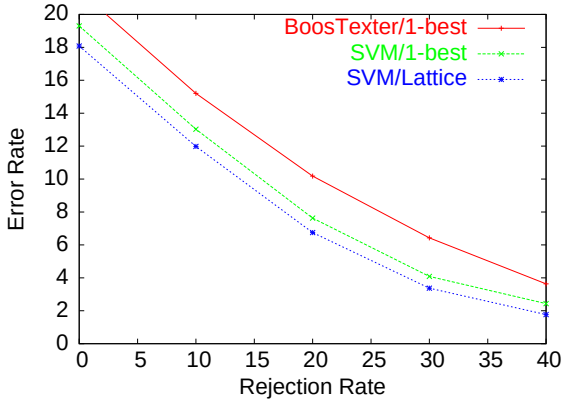


Figure 8: Classification error rate as a function of rejection rate in HMIHY 0300.

Table 7.2.2 indicates the size of the HMIHY 0300 datasets we used for training and testing. The training set is relatively large with more than 35,000 utterances, this is an extension of the one we used in our previous classification experiments with HMIHY 0300 (Cortes et al., 2003c). In our experiments, we used the n-gram rational kernels described in the previous section with $n = 3$. Thus, the feature set we used was that of all n-grams with $n \leq 3$. Table 7.2.2 indicates the total number of distinct features of this type found in the datasets. The word accuracy of the system based on the best hypothesis of the speech recognizer is 72.5%. This motivated our use of the word lattices, which may contain the correct transcription in most cases. The average number of transitions of a word lattice in this task was about 260.

Table 7.2.2 reports similar information for two other datasets, VoiceTone1, and VoiceTone2. These are more recently deployed spoken-dialog systems in different areas, e.g., VoiceTone1 is a task where users interact with a system related to health-care with a larger set of categories (97). The size of the VoiceTone1 datasets we used and the word accuracy of the recognizer (70.5%) make this task otherwise similar to HMIHY 0300. The datasets provided for VoiceTone2 are significantly smaller with a higher word error rate. The word error rate is indicative of the difficulty of classification task since a higher error rate implies a more noisy input. The average number of transitions of a word lattice in VoiceTone1 was about 210 and in VoiceTone2 about 360.

Each utterance of the dataset may be labeled with several classes. The evaluation is based on the following criterion: it is considered an error if the highest scoring class given by the classifier is none of these labels.

7.2.3 IMPLEMENTATION AND RESULTS

We used the AT&T FSM Library (Mohri et al., 2000) and the GRM Library (Allauzen et al., 2004) for the implementation of the n-gram rational kernels K_n used. We used these kernels with SVMs, using a general learning library for large-margin classification (LLAMA), which offers an optimized multi-class recombination of binary SVMs (Haffner et al., 2003). Training time took a few hours on a single processor of a 2.4GHz Intel Pentium processor Linux cluster with 2GB of memory and 512 KB cache.

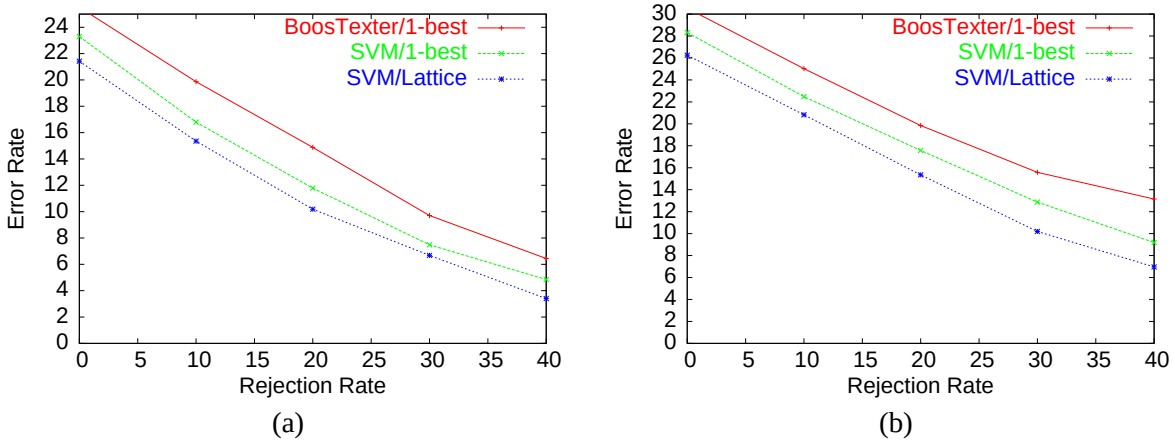


Figure 9: Classification error rate as a function of rejection rate in (a) VoiceTone1 and (b) VoiceTone2 .

In our experiments, we used the trigram kernel K_3 with a second-degree polynomial. Preliminary experiments showed that the top performance was reached for trigram kernels and that 4-gram kernels, K_4, did not significantly improve the performance. We also found that the combination of a second-degree polynomial kernel with the trigram kernel significantly improves performance over a linear classifier, but that no further improvement could be obtained with a third-degree polynomial.

We used the same kernels in the three datasets previously described and applied them to both the speech recognizer's single best hypothesis (one-best results), and to the full word lattices output by the speech recognizer. We also ran, for the sake of comparison, the BoosTexter algorithm (Schapire and Singer, 2000) on the same datasets by applying it to the one-best hypothesis. This served as a baseline for our experiments.

Figure 7.2.3 shows the result of our experiments in the HMIHY 0300 task. It gives classification error rate as a function of rejection rate (utterances for which the top score is lower than a given threshold are rejected) in HMIHY 0300 for: BoosTexter, SVM combined with our kernels when applied to the one-best hypothesis, and SVM combined with kernels applied to the full lattices.

SVM with trigram kernels applied to the one-best hypothesis leads to better classification than BoosTexter everywhere in the range of 0-40% rejection rate. The accuracy is about 2-3% absolute value better than that of BoosTexter in the range of interest for this task, which is roughly between 20% and 40% rejection rate. The results also show that the classification accuracy of SVMs combined with trigram kernels applied to word lattices is consistently better than that of SVMs applied to the one-best alone by about 1% absolute value.

Figure 7.2.3 shows the results of our experiments in the VoiceTone1 and VoiceTone2 tasks using the same techniques and comparisons. As observed previously, in many regards, VoiceTone1 is similar to the HMIHY 0300 task, and our results for VoiceTone1 are comparable to those for HMIHY 0300. The results show that the classification accuracy of SVMs combined with trigram kernels applied to word lattices is consistently better than that of BoosTexter, by more than 4% absolute value at about 20% rejection rate. They also demonstrate more clearly the benefits of the use of the word lattices for classification in this task. This advantage is even more manifest for the

VoiceTone2 task for which the speech recognition accuracy is lower. VoiceTone2 is also a harder classification task as can be seen by the comparison of the plots of Figure 7.2.3. The classification accuracy of SVMs with kernels applied to lattices is more than 6% absolute value better than that of BoosTexter near 40% rejection rate, and about 3% better than SVMs applied to the one-best hypothesis.

Thus, our experiments in spoken-dialog classification in three distinct large-vocabulary tasks demonstrates that using rational kernels with SVMs consistently leads to very competitive classifiers. They also show that their application to the full word lattices instead of the single best hypothesis output by the recognizer systematically improves classification accuracy.

8. Conclusion

We presented a general framework based on weighted transducers, rational kernels, to extend kernel methods to the analysis of variable-length sequences or more generally weighted automata. The transducer representation provides a very compact representation benefiting from existing and well-studied optimizations. It further avoids the design of special-purpose algorithms for the computation of the kernels covered by the framework of rational kernels. A single general and efficient algorithm was presented to compute effectively all rational kernels. Thus, it is sufficient to implement that algorithm and let different instances of rational kernels be given by the weighted transducers that define them. A general framework is also likely to help understand better kernels over strings or automata and their relation.

We gave the proof of several characterization results and closure properties for PDS rational kernels. These results can be used to design a complex PDS rational kernel from simpler ones or from an arbitrary weighted transducer over an appropriate semiring, or from negative definite kernels.

We also gave a study of the relation between rational kernels and several kernels or similarity measures introduced by others. Rational kernels provide a unified framework for the design of computationally efficient kernels for strings or weighted automata. The framework includes in particular pair-HMM string kernels (Durbin et al., 1998; Watkins, 1999), Haussler's convolution kernels for strings, the path kernels of Takimoto and Warmuth (2003), and other classes of string kernels introduced for computational biology. We also showed that the classical edit-distance does not define a negative definite kernel when the alphabet contains more than one symbol, a result that to our knowledge had never been stated or proved and that can guide the study of kernels for strings in computational biology and other similar applications.

Our experiments in several different large-vocabulary spoken-dialog tasks show that rational kernels can be combined with SVMs to form powerful classifiers and demonstrate the benefits of the use of kernels applied to weighted automata. There are many other rational kernels such as complex gappy n-gram kernels that could be explored and that perhaps could further improve classification accuracy in such experiments. We present elsewhere new rational kernels exploiting higher-order moments of the distribution of the counts of sequences, *moment kernels*, and report the results of our experiments on the same tasks which demonstrate a consistent gain in classification accuracy (Cortes and Mohri, 2004). Rational kernels can be used in a similar way in many other natural language processing, speech processing, and bioinformatics tasks.

Acknowledgments

We thank Allen Van Gelder, David Haussler, Risi Kondor, Alex Smola, and Manfred Warmuth for discussions about this work. We are also grateful to our colleagues of the AT&T HMIHY and VoiceTone teams for making available the data we used in our experiments.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. A General Weighted Grammar Library. In *Proceedings of the Ninth International Conference on Automata (CIAA 2004)*, Kingston, Ontario, Canada, July 2004. URL `http://www.research.att.com/sw/tools/grm`.

Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag: Berlin-New York, 1984.

Jean Berstel. *Transductions and Context-Free Languages*. Teubner Studienbucher: Stuttgart, 1979.

Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*. Springer-Verlag: Berlin-New York, 1988.

Bernhard E. Boser, Isabelle Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop of Computational Learning Theory*, volume 5, pages 144–152, Pittsburg, 1992. ACM.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Lattice Kernels for Spoken Dialog Classification. In *Proceedings ICASSP'03*, Hong Kong, 2003a.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Positive Definite Rational Kernels. In *Proceedings of The 16th Annual Conference on Computational Learning Theory (COLT 2003)*, volume 2777 of *Lecture Notes in Computer Science*, pages 41–56, Washington D.C., August 2003b. Springer, Heidelberg, Germany.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels. In *Advances in Neural Information Processing Systems (NIPS 2002)*, volume 15, Vancouver, Canada, March 2003c. MIT Press.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Weighted Automata Kernels – General Framework and Algorithms. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Eurospeech '03), Special Session Advanced Machine Learning Algorithms for Speech and Language Processing*, Geneva, Switzerland, September 2003d.

Corinna Cortes and Mehryar Mohri. Distribution Kernels Based on Moments of Counts. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, Banff, Alberta, Canada, July 2004.

Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3): 273–297, 1995.

Jack J. Dongarra, Jim R. Bunch, Cleve B. Moler, and G. W. Stewart. *LINPACK User's Guide*. SIAM Publications, 1979.

Richard Durbin, Sean Eddy, Anders Krogh, , and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK, 1998.

Samuel Eilenberg. *Automata, Languages and Machines*, volume A-B. Academic Press, 1974.

Patrick Haffner, Gokhan Tur, and Jeremy Wright. Optimizing SVMs for complex Call Classification. In *Proceedings ICASSP'03*, 2003.

David Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1986.

Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, 1976.

Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch String Kernels for SVM Protein Classification. In *NIPS 2002*, Vancouver, Canada, March 2003. MIT Press.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10:707–710, 1966.

Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS 2000*, pages 563–569. MIT Press, 2001.

Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

Mehryar Mohri. Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982, 2003.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended finite state models of language*, Budapest, Hungary, 1996. John Wiley and Sons, Chichester.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. The Design Principles of a Weighted Finite-State Transducer Library. *Theoretical Computer Science*, 231:17–32, January 2000. URL `http://www.research.att.com/sw/tools/fsm`.

Fernando C. N. Pereira and Michael D. Riley. Speech recognition by composition of weighted finite automata. In *Finite-State Language Processing*, pages 431–453. MIT Press, Cambridge, Massachusetts, 1997.

Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag: New York, 1978.

Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

Bernhard Schölkopf. The Kernel Trick for Distances. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS 2001*, pages 301–307. MIT Press, 2001.

Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press: Cambridge, MA, 2002.

Izhak Shafran, Michael Riley, and Mehryar Mohri. Voice Signatures. In *Proceedings of The 8th IEEE Automatic Speech Recognition and Understanding Workshop (ASRU 2003)*, St. Thomas, U.S. Virgin Islands, November 2003.

Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *The Journal of Machine Learning Research (JMLR)*, 4:773 – 818, 2003.

Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

Chris Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Royal Holloway, University of London, 1999.

Journal of Machine Learning Research 5 (2004) 1063–1088 Submitted 3/02; Revised 1/04; Published 8/04

Reinforcement Learning with Factored States and Actions

Brian Sallans SALLANS@OEFAI.AT
Austrian Research Institute for Artificial Intelligence
Freyung 6/6
A-1010 Vienna, Austria

Geoffrey E. Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
Toronto, Canada

Editor: Sridhar Mahadevan

Abstract

A novel approximation method is presented for approximating the value function and selecting good actions for Markov decision processes with large state and action spaces. The method approximates state-action values as negative free energies in an undirected graphical model called a product of experts. The model parameters can be learned efficiently because values and derivatives can be efficiently computed for a product of experts. Actions can be found even in large factored action spaces by the use of Markov chain Monte Carlo sampling. Simulation results show that the product of experts approximation can be used to solve large problems. In one simulation it is used to find actions in action spaces of size 2^{40}.

Keywords: product of experts, Boltzmann machine, reinforcement learning, factored actions

1. Introduction

An agent must be able to deal with high-dimensional and uncertain states and actions in order to operate in a complex environment. In this paper we focus on two related problems: estimating the value of a state-action pair in large state and action spaces; and selecting good actions given these estimates. Our approach is to borrow techniques from the graphical modeling literature and apply them to the problems of value estimation and action selection.

Inferring the state of the agent's environment from noisy observations has been a popular subject of study in the engineering, artificial intelligence and machine learning communities. One formalism is the graphical model (Cowell et al., 1999). A graphical model represents the distribution of observed data with a probabilistic model. The graphical representation of the model indicates which variables can be assumed to be conditionally independent. Given observations of some variables, inferring the distribution over unobserved (or hidden) variables is of paramount importance in using and learning the parameters of these models. Exact and approximate inference algorithms have been and still are intensely studied in the graphical models and engineering literatures (Kalman, 1960; Neal, 1993; Jordan et al., 1999; Cowell et al., 1999).

Acting on certain or uncertain information has been studied in a different body of literature. Reinforcement learning involves learning how to act so as to maximize a reward signal given samples

of sequences of state-action pairs and rewards from the environment (Sutton and Barto, 1998). It has been closely linked to Markov decision processes and stochastic dynamic programming (see for example Sutton, 1988; Bertsekas and Tsitsiklis, 1996). There has been work on reinforcement learning with large state spaces, state uncertainty and partial observability (see for example Bertsekas and Tsitsiklis, 1996; Jaakkola et al., 1995). In particular there are exact dynamic programming methods for solving fully and partially observable Markov decision processes (see Lovejoy, 1991, for an overview). There are also approximate methods for dealing with real-valued state and action variables (see for example Baird and Klopf, 1993; Sutton, 1996; Santamaria et al., 1998).

Recently, techniques from the graphical models literature have started to come together with those from the planning and reinforcement-learning community. The result has been new algorithms and methods for learning about decision processes and making decisions under uncertainty in complex and noisy environments (see for example Boutilier and Poole, 1996; McAllester and Singh, 1999; Thrun, 2000; Sallans, 2000; Rodriguez et al., 2000; Poupart and Boutilier, 2001).

In this article, we propose to make use of techniques from graphical models and approximate inference to approximate the values of and select actions for large Markov decision processes. The value function approximator is based on an undirected graphical model called a product of experts (PoE). The value of a state-action pair is modeled as the negative free energy of the state-action under the product model.

Computing the free energy is tractable for a product of experts model. However, computing the resulting distribution over actions is not. Given a value function expressed as a product of experts, actions can be found by Markov chain Monte Carlo (MCMC) sampling. As with any sampling scheme, it is possible that action sampling will perform poorly, especially as the action space becomes large. There are no theoretical guarantees as to the effectiveness of sampling for short periods of time.

The advantage of using MCMC sampling for action selection is that there has been a concerted effort put into making sampling methods work well in large state (or in our case, action) spaces. It is also possible to use this technique to approximate value functions over real-valued state and action spaces, or mixtures of discrete and real-valued variables, and to make use of MCMC sampling methods designed for continuous spaces to do action selection. It is an empirical question whether or not action sampling works well for specific problems.

Our technique uses methods from reinforcement learning and from products of experts modeling. We therefore include a short review of the Markov decision process formalism, reinforcement learning, and products of experts. For clarity, we will focus on one particular kind of product of experts model: the restricted Boltzmann machine.

We then describe the method, which uses a product of experts network as a novel value function approximator. We demonstrate the properties of the PoE approximation on two tasks, including an action-selection task with a 40-bit action space. We conclude with some discussion of the approximation method and possible future research.

2. Markov Decision Processes

An agent interacting with the environment can be modeled as a Markov decision process (MDP) (Bellman, 1957b). The task of learning which action to perform based on reward is formalized by reinforcement learning. Reinforcement learning in MDPs is a much studied problem. See for example Sutton and Barto (1998).

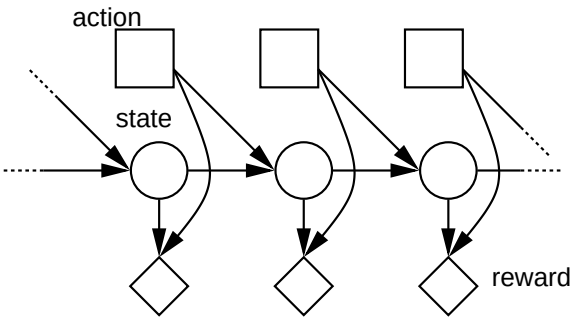


Figure 1: A Markov decision process. Circles indicate visible variables, and squares indicate actions. The state is dependent on the previous state and action, and the reward depends on the current state and action.

If the sets of states and actions are finite, then the problem is called a finite MDP. Much of the theoretical work done in reinforcement learning has focused on the finite case, and we focus on finite MDPs in this article.

Formally, an MDP consists of

- A set of states $\mathcal{S}$, and actions $\mathcal{A}$,
- An initial state s^0 or distribution $P(s^0)$,
- A transition distribution $P(s^{t+1}|s^t,a^t)$, $s^t,s^{t+1}\in\mathcal{S}$, $a^t\in\mathcal{A}$, and
- A reward distribution $P(r^t|s^t,a^t)$, $s^t\in\mathcal{S}$, $r^t\in\mathbb{R}$, $a^t\in\mathcal{A}$.

In the above, t indexes the time step, which ranges over a discrete set of points in time. We will denote the transition probability $Pr(s^{t+1}=j\,|s^t=i,a^t=a\,)$ by $\mathbf{P}_{ij}(a)$. We will also denote the expected immediate reward received by executing action a in state i by

$$r_i(a)=\left\langle r^t\,\middle|s^t=i,a^t=a\,\right\rangle_{P(r^t|s^t,a^t)},$$

where $\langle\cdot\rangle_P$ denotes expectation with respect to distribution P. Bold-face text denotes vectors or matrices.

The goal of solving an MDP is to find a *policy* which maximizes the total expected reward received over the course of the task. A policy tells the learning agent what action to take for each possible state. It is a mapping π from states to actions or distributions over actions. We will focus on *stationary* policies, in which the same mapping is used at every point in time.

The expected discounted *return* for a policy π is defined as the sum of discounted rewards that is expected when following policy π:

$$\left\langle R^t\right\rangle_\pi \;=\; \left\langle r^t+\gamma r^{t+1}+\gamma^2 r^{t+2}+\ldots\right\rangle_\pi \;=\; \left\langle\sum_{k=0}^{\infty}\gamma^k r^{t+k}\right\rangle_\pi,$$

where t is the current time, and $\pi(s,a)$ is the probability of selecting action a in state s. Note that the discounting is required to ensure that the sum of infinite (discounted) rewards is finite, so that

the quantity to be optimized is well-defined; the discount factor $\gamma \in [0,1)$. The expectation is taken with respect to the policy π, the initial state distribution $P(s^0)$, the transition distribution $\mathbf{P}_{ij}(a)$, and the reward distribution $P(r|s,a)$.

To solve an MDP, we must find a policy that produces the greatest expected return. With knowledge of transition probabilities $\mathbf{P}_{ij}(a)$ and expected immediate rewards $r_i(a)$, and given a stochastic policy π, we can calculate the expected discounted return after taking the action a from the current state s and following policy π thereafter:

$$\begin{aligned} Q^\pi(s,a) &= \left\langle \sum_{k=0}^{\infty} \gamma^k r^{t+k} \,\middle|\, s^t = s, a^t = a \right\rangle_\pi \\ &= \left\langle r^t + \gamma \sum_{k=0}^{\infty} \gamma^k r^{t+k+1} \,\middle|\, s^t = s, a^t = a \right\rangle_\pi \\ &= \sum_j \mathbf{P}_{sj}(a) \left[r_s(a) + \sum_b \pi(j,b) \gamma Q^\pi(j,b) \right]. \end{aligned} \tag{1}$$

Here t denotes the current time. The function Q^π is called the *action-value function* for policy π. The action-value function tells the agent the expected return that can be achieved by starting from any state, executing an action, and then following policy π.

Equation 1 is often called the Bellman equations for Q^π. It is a set of $|\mathcal{S}| \times |\mathcal{A}|$ linear equations (one for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$). The set of coupled equations can be solved for the unknown values $Q^\pi(s,a)$. In particular, given some arbitrary initialization Q_0^π, we can use the following iterative update:

$$Q_{k+1}^\pi(s,a) = \sum_j \mathbf{P}_{sj}(a) \left[r_s(a) + \sum_b \pi(j,b) \gamma Q_k^\pi(j,b) \right] \tag{2}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The iteration converges to the unique fixed-point Q^π as $k \to \infty$. This technique is called iterative policy evaluation.

The class of MDPs is a restricted but important class of problems. By assuming that a problem is Markov, we can ignore the history of the process, and thereby prevent an exponential increase in the size of the domain of the policy (Howard, 1960). The Markov assumption underlies a large proportion of control theory, machine learning and signal processing including Kalman filters (Kalman, 1960), hidden Markov models (HMMs) (Rabiner and Juang, 1986), two-time-slice dynamic Bayesian networks (Dean and Kanazawa, 1989), dynamic programming (DP) (Bellman, 1957a) and temporal difference learning (TD) (Sutton, 1988).

When the states or actions are composed of sets of variables, we will refer to them as "factored" states or actions. It is common for problems with large state or action spaces to have states and actions expressed in a "factored" form. There has been a lot of research and discussion about the problems of dealing with large state spaces represented in factored form (see for example Bertsekas and Tsitsiklis, 1996). There has been comparatively little on dealing with large action spaces or factored actions (Dean et al., 1998; Meuleau et al., 1998; Peshkin et al., 1999; Guestrin et al., 2002). In practice action spaces tend to be very large. For example consider the activation of large numbers of muscles, or the simultaneous actions of all of the players on a football field. The state or action space could also be continuous (Baird and Klopf, 1993; Santamaria et al., 1998).

3. Temporal Difference Learning

Temporal difference (TD) learning (Sutton, 1988) addresses the problem of predicting time-delayed rewards. We can view TD as performing approximate dynamic programming to compute future reward. Because they use a value function as an estimate of future performance instead of sampled rewards, TD algorithms trade off bias against variance in estimates of future reward. The nature of the approximation, this tradeoff, and the convergence of TD algorithms have been the subjects of much study.

Temporal difference algorithms can be used to estimate the value of states and actions. TD techniques update the value estimate for states and actions as they are visited and executed. Backing up the values of states only as they are visited gives rise to a number of TD update rules. The SARSA algorithm computes the action-value function of the current policy (Rummery and Niranjan, 1994; Sutton, 1996):

$$Q(s^t, a^t) \leftarrow (1-\kappa)Q(s^t, a^t) + \kappa\left[r^t + \gamma Q(s^{t+1}, a^{t+1})\right], \tag{3}$$

where κ is a learning rate.

This update can be viewed as a Monte Carlo approximation to the update from Eq.(2). The name derives from the fact that the update depends on the set of values $\{s^t, a^t, r^t, s^{t+1}, a^{t+1}\}$. SARSA computes the expected return conditioned on a state-action pair. The update is designed to move the estimated value function closer to a "bootstrapped" Monte Carlo estimate. If κ is reduced over time in the appropriate manner, and all states continue to be visited an infinite number of times, then this algorithm will converge to the value function of the current policy (Singh et al., 2000).

Given the state-action value function, a greedy policy with respect to the value function can be found by maximizing over possible actions in each state:

$$\pi(s) = \operatorname*{argmax}_a Q^\pi(s, a).$$

Note that this involves an explicit maximization over actions. When the action space is large or continuous, this maximization will become difficult.

The optimal value function and policy can be found using SARSA, by combining value function estimation (Eq.3) with policies which become greedy with respect to the value function, in the limit of infinite time (i.e. an infinite number of value function updates, with all states and actions continuing to be visited/executed). See Singh et al. (2000) for a proof of convergence and conditions on the policies.

4. Function Approximation

In many problems there are too many states and actions to represent the action-value function as a state-action table. One alternative is to use function approximation. If the approximation is differentiable with respect to its parameters, the parameters can be learned by trying to minimize the TD error. The TD error is defined as

$$E_{\mathrm{TD}}(\mathbf{s}^t, \mathbf{a}^t) = \left[r^t + \gamma Q(\mathbf{s}^{t+1}, \mathbf{a}^{t+1})\right] - Q(\mathbf{s}^t, \mathbf{a}^t). \tag{4}$$

Consider an approximate value function $Q(\mathbf{s}, \mathbf{a}; \theta)$ parameterized by parameter θ. The update rule for the parameter θ is given by

$$\Delta\theta = \lambda E_{\mathrm{TD}}(\mathbf{s}, \mathbf{a}) \nabla_\theta Q(\mathbf{s}, \mathbf{a}; \theta), \tag{5}$$

where λ is a learning rate. This is the approach taken by Bertsekas and Tsitsiklis (1996) among others. Although this approach can work in some cases, there are in general no guarantees of convergence to a specific approximation, or guarantees of the accuracy of the approximation if it does converge.

5. Boltzmann Machines and Products of Experts

We will use a product of experts model to approximate the values of states and actions in a Markov decision process (Hinton, 1999, 2002). Products of experts are probabilistic models that combine simpler models by multiplying their distributions together.

In this section we will focus on a particular kind of product of experts, called a restricted Boltzmann machine (Smolensky, 1986; Freund and Haussler, 1992; Hinton, 2002). This case is interesting because inference and learning in this model has been intensely studied (Ackley et al., 1985; Hinton and Sejnowski, 1986; Smolensky, 1986; Freund and Haussler, 1992; Hinton, 2002), and the binary values are relatively easy to interpret.

Boltzmann machines are undirected models. That means that the model specifies joint probabilities, rather than conditional probabilities. Directed graphical models, for example Bayesian networks, have also been used in conjunction with value function approximation, where the Bayesian network encodes a compact model of the environment (see for example Boutilier and Poole, 1996; Boutilier et al., 2000; Rodriguez et al., 2000; Sallans, 2000; Thrun, 2000). They have also been used to directly encode utilities (Boutilier et al., 1999, 2001).

The free energy of a directed model could also be used to encode an approximate value function. However, unlike with product models, the computation of the free energy and its derivatives is generally intractable for directed models. This is because inference in the model is not tractable. Research in approximate inference techniques for directed models is an important area of current research.

In a directed model, it is also intractable to compute the conditional distribution over actions given states, as with the product of experts models. It would be possible to combine an approximate inference technique with a directed model to approximate the value, and also use the approximate inference technique to sample actions from the network. The result would have the flavor of an actor-critic network, where the free energy of the directed model plays the role of the critic, and the approximate inference technique plays the role of the actor. The advantage would be that the distribution over actions could be evaluated, rather than just sampled from as with the product of experts. The disadvantage is that we would use an approximation not just for action selection, but also to compute the free energy and its derivatives.

We begin the next section with a discussion of the general Boltzmann machine (Ackley et al., 1985) and review some necessary concepts such as energy and free energy, the Boltzmann distribution, and Markov chain Monte Carlo sampling. We then discuss the restricted Boltzmann machine. For completeness, we include some derivations, but defer them to Appendix A. The reader is directed to Hertz et al. (1991), chapter 7, for a more in-depth introduction to the Boltzmann machine. Finally, we discuss more general products of experts.

5.1 Boltzmann Machines

A Boltzmann machine is an undirected graphical model. The nodes represent binary random variables that have values of 1 or 0 and the weighted edges represent pairwise symmetric interactions

between the variables.[1] The nodes are usually divided into two disjoint subsets, the "visible" variables, $\mathbf{V}$ and the "hidden" variables, $\mathbf{H}$. An assignment of binary values to the visible or hidden variables will be denoted by $\mathbf{v}$ or $\mathbf{h}$ and the binary value of an individual visible or hidden variable, V_i or H_k, will be denoted by v_i or h_k. The symmetric weight between node i and node k is w_{ik}. In a general Boltzmann machine, weights can occur between any pair of nodes.

The weights determine the "energy" of every possible joint configuration of the visible and hidden variables:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i,k} w_{ik} v_i h_k - \sum_{i<j} w_{ij} v_i v_j - \sum_{k<m} w_{km} h_k h_m,$$

where i and j are indices over visible variables and k and m are indices over hidden variables. The energies of the joint configurations determine their equilibrium probabilities via the Boltzmann distribution:

$$P(\mathbf{v},\mathbf{h}) = \frac{\exp(-E(\mathbf{v},\mathbf{h}))}{\sum_{\widehat{\mathbf{v}},\widehat{\mathbf{h}}} \exp(-E(\widehat{\mathbf{v}},\widehat{\mathbf{h}}))},$$

where $\widehat{\mathbf{v}},\widehat{\mathbf{h}}$ indexes all joint configurations of the visible and hidden variables.

The probability distribution over the visible variables can be obtained by summing over all possible configurations of the hidden variables:

$$\exp(-F(\mathbf{v})) = \sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h})), \tag{6}$$

$$P(\mathbf{v}) = \frac{\exp(-F(\mathbf{v}))}{\sum_{\widehat{\mathbf{v}}} \exp(-F(\widehat{\mathbf{v}}))}.$$

$F(\mathbf{v})$ is called the "equilibrium free energy" of $\mathbf{v}$. It is the minimum of the "variational free energy" of $\mathbf{v}$ which can be expressed as an expected energy minus an entropy:

$$F_q(\mathbf{v}) = \sum_{\mathbf{h}} q(\mathbf{h}) E(\mathbf{v},\mathbf{h}) + \sum_{\mathbf{h}} q(\mathbf{h}) \log q(\mathbf{h}), \tag{7}$$

where q is any distribution over all possible configurations of the hidden units. To make the first term in Eq.(7) low, the distribution q should put a lot of mass on hidden configurations for which $E(\mathbf{v},\mathbf{h})$ is low, but to make the second term low, the q distribution should have high entropy. The optimal trade-off between these two terms is the Boltzmann distribution in which $P(\mathbf{h}|\mathbf{v}) \propto \exp(-E(\mathbf{v},\mathbf{h}))$:

$$P(\mathbf{h}|\mathbf{v}) = \frac{\exp(-E(\mathbf{v},\mathbf{h}))}{\sum_{\widehat{\mathbf{h}}} \exp(-E(\mathbf{v},\widehat{\mathbf{h}}))}. \tag{8}$$

This is the posterior distribution over the hidden variables, given the visible variables. Using this distribution, the variational free energy defined by Eq.(7) is equal to the equilibrium free energy in Eq.(6):

$$F(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) E(\mathbf{v},\mathbf{h}) + \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) \log P(\mathbf{h}|\mathbf{v}). \tag{9}$$

1. We will assume that the variables can take on values of 1 or 0. Alternatively, the Boltzmann machine can be formulated with values of ± 1. We also omit biases on variables, which can be incorporated into the weights by adding a visible variable which always has a value of one. Weights from this "always on" variable act as biases to the other variables.

In Appendix A, we show that the equilibrium free energy can be written either as in Eq.(6) or as in Eq.(9) (see Appendix A, Eq.11).

One important property of Boltzmann machines is that, with enough hidden variables, a Boltzmann machine with finite weights is capable of representing any "soft" distribution over the visible variables, where "soft" means that the distribution does not contain any probabilities of 1 or 0. Another important property is that there is a simple, linear relationship between $F(\mathbf{v})$ and each of the weights in the network (Ackley et al., 1985). For a weight between a visible and a hidden unit this relationship is

$$\frac{\partial F(\mathbf{v})}{\partial w_{ik}} = -v_i \left\langle h_k \right\rangle_{P(h_k|\mathbf{v})},$$

where the angle brackets denote the expectation of h_k under the distribution $P(h_k|\mathbf{v})$. At first sight, it is surprising that this derivative is so simple because changing w_{ik} will clearly change the equilibrium distribution over the hidden configurations. However, to first order, the changes in this equilibrium distribution have no effect on $F(\mathbf{v})$ because the equilibrium distribution $P(\mathbf{h}|\mathbf{v})$ is the distribution for which $F(\mathbf{v})$ is minimal, so the derivatives of $F(\mathbf{v})$ *w.r.t.* the probabilities in the distribution are all zero (see Appendix A).

For a general Boltzmann machine, it is computationally intractable to compute the equilibrium distribution over the hidden variables given a particular configuration, $\mathbf{v}$, of the visible variables. (Cooper, 1990). However, values can be sampled from the equilibrium distribution by using a Markov chain Monte Carlo method. Once the Markov chain reaches equilibrium (in other words, all information about the initialization has been lost), hidden configurations are sampled according to the Boltzmann distribution (Eq.8).

One sampling method is called the Gibbs sampler (Geman and Geman, 1984). Given a fixed configuration $\mathbf{v}$ of the visible variables, the Gibbs sampler proceeds as follows.

1. Initialize all the hidden variables with arbitrary binary values: $h_1^0, \ldots h_k^0, \ldots, h_K^0$.
2. Repeat the following until convergence:
   In each iteration $t = 1, 2, \ldots$, and for each random variable h_k:
   (a) Compute the energy $E_1 = E(\mathbf{v}, h_k = 1, \{h_m = h_m^{t-1} : m \neq k\})$.
   (b) Compute the energy $E_0 = E(\mathbf{v}, h_k = 0, \{h_m = h_m^{t-1} : m \neq k\})$.
   (c) Set $h_k = 1$ with probability $\exp(-E_1)/(\exp(-E_0) + \exp(-E_1))$
   and set $h_k = 0$ with probability $\exp(-E_0)/(\exp(-E_0) + \exp(-E_1))$.

This procedure should be repeated until the Markov chain converges to its stationary distribution which is given by Eq.(8). Assessing whether or not convergence has occurred is not trivial, and will not be discussed here.

The Gibbs sampler is only one possible sampling technique. There are many others, including the Metropolis-Hastings algorithm (Metropolis et al., 1953), and hybrid Monte Carlo algorithms (Duane et al., 1987). The reader is directed to Neal (1993) for a review of Markov chain Monte Carlo methods.

The difficulty of computing the posterior over the hidden variables in a general Boltzmann machine makes it unsuitable for our purposes, because it means that the free energy of a visible vector

can not be easily evaluated. However, a restricted class of Boltzmann machines (Smolensky, 1986; Freund and Haussler, 1992; Hinton, 1999, 2002) is more useful to us. In a restricted Boltzmann machine there are no hidden-hidden or visible-visible connections but any hidden-visible connection is allowed. The connectivity of a restricted Boltzmann machine therefore forms a bipartite graph.

In a restricted Boltzmann machine the posterior distribution over the hidden variables factors into the product of the posteriors over each of the individual hidden units (Freund and Haussler, 1992) (see Appendix A for a derivation):

$$P(\mathbf{h}|\mathbf{v}) = \prod_k P(h_k|\mathbf{v}).$$

The posterior over hidden variables can therefore be computed efficiently, because each individual hidden-unit posterior is tractable:

$$P(h_k = 1|\mathbf{v}) = \frac{1}{1+\exp(-\sum_i v_i w_{ik})}.$$

This is crucial, because it allows for efficient computation of the equilibrium free energy $F(\mathbf{v})$ and of its derivatives with respect to the weights.

After a Boltzmann machine has learned to model a distribution over the visible variables, it can be used to complete a visible vector that is only partially specified. If, for example, one half of the visible vector represents a state and the other half represents an action, the Boltzmann machine defines a probability distribution over actions for each given state. Moreover, Gibbs sampling in the space of actions can be used to pick actions according to this distribution (see below).

In summary, restricted Boltzmann machines have the properties that we require for using negative free energies to approximate Q-values: The free energy and its derivatives can be efficiently computed; and, given a state, Gibbs sampling can be used to sample actions according a Boltzmann distribution in the free energy.

5.2 Products of Experts

Restricted Boltzmann machines are only one example of a class of models called products of experts (Hinton, 2002). Products of experts combine simple probabilistic models by multiplying their probability distributions. In the case of restricted Boltzmann machines, the individual "experts" are stochastic binary hidden variables. Products of experts share the useful properties discussed above for restricted Boltzmann machines. A product of experts model defines a free energy whose value and its derivative can be efficiently computed; and instantiations of the random variables can be sampled according to the Boltzmann distribution. Products of experts can include more complex individual experts than binary variables. Examples of product models include products of hidden Markov models (Brown and Hinton, 2001) and products of Gaussian mixtures (Hinton, 2002). Notice that in each case the individual experts (hidden Markov models and Gaussian mixtures) are themselves tractable, meaning that the posterior distribution over an individual expert's hidden variables, and derivatives of the free energy with respect to the parameters, can be computed tractably. This is all that is required for the entire product model to be tractable. Although we focus on restricted Boltzmann machines in this work, other products of experts models could also be used, for example, to model real-valued variables.

6. The Product of Experts as Function Approximator

Consider a product of experts model, where the visible variables are state and action variables. The free energy allows a PoE to act as a function approximator, in the following sense. For any input (instantiation of the visible variables), the output of the function approximator is taken to be the free energy. With no hidden variables, the output is simply the energy. For a Boltzmann machine, this is similar to a linear neural network with no hidden units. With hidden variables, the Boltzmann machine is similar to a neural network with hidden units. However, unlike traditional neural networks, having probabilistic semantics attached to the model allows us to (at least approximately) sample variables according to the Boltzmann distribution. This is ideal for value function approximation, because we can sample actions according to a Boltzmann exploration policy, conditioned on settings of the state variables, even in large action spaces for which actually computing the Boltzmann distribution would be intractable. To do this, we have to create a correspondence between the value of a state-action pair, and its negative free energy under the Boltzmann machine model.

We create this correspondence using the parameter update rule for reinforcement learning with function approximation (Eq.5). The parameters of the PoE model are updated to try to reduce the temporal-difference error (Eq.4). By reducing the temporal-difference error, we make the value approximated by the product of experts closer to the correct value.

Once the negative free energy under the PoE model approximates the value, we use MCMC sampling to select actions. After training, the probability of sampling an action from the product of experts while holding the state fixed is given by the Boltzmann distribution:

$$P(\mathbf{a}|\mathbf{s}) = \frac{e^{-F(\mathbf{s},\mathbf{a})/T}}{Z} \approx \frac{e^{Q(\mathbf{s},\mathbf{a})/T}}{Z},$$

where Z is a normalizing constant, and T is the exploration temperature. Samples can be selected at a particular exploration temperature by dividing the free energy by this temperature.

Intuitively, good actions will become more probable under the model, and bad actions will become less probable under the model. Although finding optimal actions would still be difficult for large problems, selecting an action with a probability that is approximately the probability under the Boltzmann distribution can normally be done with a small number of iterations of MCMC sampling (and could include simulated annealing). In principle, if we let the MCMC sampling converge to the equilibrium distribution, we could draw unbiased samples from the Boltzmann exploration policy at a given temperature. In particular we can select actions according to a Boltzmann exploration policy that may be intractable to compute explicitly, because normalization would require summing over an exponential number of actions. In practice, we only sample for a short period of time. It should be noted that this "brief" sampling comes with no performance guarantees, and may be problematic in large action spaces. However, we can easily incorporate improvements in sampling techniques to improve performance in large discrete and real-valued action spaces.

6.1 Restricted Boltzmann Machines

Here we detail the approximation architecture for the specific example of a restricted Boltzmann machine. We approximate the value function of an MDP with the negative free energy of the restricted Boltzmann machine (Eq.6). The state and action variables will be assumed to be discrete, and will be represented by the visible binary variables of the restricted Boltzmann machine.

In the following section, the number of binary state variables will be denoted by N; the number of binary action variables by M; and the number of hidden variables by K. We will represent a discrete multinomial state or action variable of arity J by using a "one-of-J" set of binary variables which are constrained so that exactly one of them is unity, and the rest are zero.

We will use Gibbs sampling to select actions. To take the multinomial restriction into account, the sampling method must be modified. Specifically, instead of sampling each variable in sequence, we will sample simultaneously over a group of J variables that represents a multinomial variable of arity J. This is done by first computing the energy of each instantiation where one of the group takes on a value of unity, and the others are zero. Let F_i be the free energy of the instantiation where $s_i = 1$. This instantiation is selected as the new sample with probability $e^{-F_i}/[\sum_j e^{-F_j}]$.

The restricted Boltzmann machine is shown in Figure 2(a). We use s_i to denote the i^{th} state variable and a_j to denote the j^{th} action variable. We will denote the binary hidden variables by h_k. Weights between hidden and state variables will be denoted w_{ik}, and weights between hidden and action variables will be denoted u_{jk} (Figure 2 (b)).

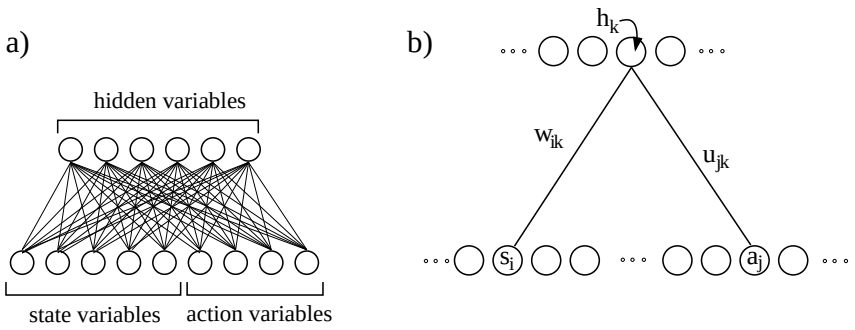


Figure 2: a) The restricted Boltzmann machine. The estimated action-value of a setting of the state and action variables is found by holding these variables fixed and computing the negative free energy of the model. Actions are selected by holding the state variables fixed and sampling from the action variables.
b) The state variables are denoted s_i, the actions a_j and the hidden variables h_k. A hidden-state weight is denoted by w_{ik} and a hidden-action weight by u_{jk}.

In the following, keep in mind that state variables are always held fixed, and the actions are always sampled such that any one-of-J multinomial restrictions are respected. Given these restrictions, we can ignore the fact that the binary vector may represent the values of a set of multinomial variables. The representation of the free energy is the same as in the binary case.[2]

For a state $\mathbf{s} = \{s_i : i \in \{1,...,N\}\}$ and an action $\mathbf{a} = \{a_j : j \in \{1,...,M\}\}$, the free energy is given by Eq.(6), restated here in terms of state, action, and hidden variables:

$$\begin{aligned} F(\mathbf{s},\mathbf{a}) &= -\sum_{k=1}^{K}\left(\sum_{i=1}^{N}(w_{ik}s_i\langle h_k\rangle)+\sum_{j=1}^{M}(u_{jk}a_j\langle h_k\rangle)\right) \\ &\quad +\sum_{k=1}^{K}\langle h_k\rangle\log\langle h_k\rangle+(1-\langle h_k\rangle)\log(1-\langle h_k\rangle). \end{aligned} \tag{10}$$

2. This is equivalent to the Potts multinomial model formulation (Potts, 1952).

The expected value of the hidden variable $\langle h_k \rangle$ is given by

$$\langle h_k \rangle = \sigma \left(\sum_{i=1}^{N} w_{ik} s_i + \sum_{j=1}^{M} u_{jk} a_j \right),$$

where $\sigma(x) = 1/(1+e^{-x})$ denotes the logistic function. The first line of Eq.(10) correspond to an expected energy, and the second to the negative entropy of the distribution over the hidden variables given the data. The value of a state-action pair is approximated by the negative free energy

$$\widehat{Q}(\mathbf{s}, \mathbf{a}) = -F(\mathbf{s}, \mathbf{a}).$$

6.2 Learning Model Parameters

The model parameters are adjusted so that the negative free energy of a state-action pair under the product model approximates its action-value. We will use the temporal difference update rule SARSA (Eq.3). The temporal-difference error quantifies the inconsistency between the value of a state-action pair and the discounted value of the next state-action pair, taking the immediate reinforcement into account.

The SARSA update is a parameter update rule where the target for input $(\mathbf{s}^t, \mathbf{a}^t)$ is $r^t + \gamma \widehat{Q}(\mathbf{s}^{t+1}, \mathbf{a}^{t+1})$. The update for w_{ik} is given by

$$\Delta\, w_{ik} \;\propto\; \left(r^t + \gamma \widehat{Q}(\mathbf{s}^{t+1}, \mathbf{a}^{t+1}) - \widehat{Q}(\mathbf{s}^t, \mathbf{a}^t) \right) s_i^t \left\langle h_k^t \right\rangle .$$

The other weights are updated similarly:

$$\Delta\, u_{jk} \;\propto\; \left(r^t + \gamma \widehat{Q}(\mathbf{s}^{t+1}, \mathbf{a}^{t+1}) - \widehat{Q}(\mathbf{s}^t, \mathbf{a}^t) \right) a_j^t \left\langle h_k^t \right\rangle .$$

This is found by plugging the derivative of the free energy with respect to a parameter into the update rule (Eq.5). Although there is no proof of convergence in general for this learning rule, it can work well in practice even though it ignores the effect of changes in parameters on $\widehat{Q}(\mathbf{s}^{t+1}, \mathbf{a}^{t+1})$. It is possible to derive update rules that use the actual gradient. See for example Baird and Moore (1999).

6.3 Exploration

Given that we can select actions according to their value, we still have to decide on an exploration strategy. One common action selection scheme is Boltzmann exploration. The probability of selecting an action is proportional to $e^{\widehat{Q}(s,a)/T}$. It can move from exploration to exploitation by adjusting the "temperature" parameter T. This is ideal for our product of experts representation, because it is natural to sample actions according to this distribution.

Another possible selection scheme is ε-greedy, where the optimal action is selected with probability $(1-\varepsilon)$ and a random action is selected with probability ε. The exploration probability ε can be reduced over time, to move the learner from exploration to exploitation.

If the SARSA update rule is used with Boltzmann exploration then samples from the Boltzmann distribution at the current temperature are sufficient. This is what we do in our experimental section. If ε-greedy is used we must also evaluate $\max_{\mathbf{a}} \widehat{Q}(\mathbf{s}, \mathbf{a})$. This can be approximated by sampling at a low temperature. To improve the approximation, the temperature can be initialized to a high value and lowered during the course of sampling.

7. Simulation Results

To test the approximation we introduce two tasks: the large-action task and the blockers task. The former involves no delayed rewards, and is designed to test action sampling in a large action space. The latter is smaller, but tests learning with delayed reward. We compare performance against two competing algorithms: a direct policy algorithm and a feed-forward neural network with simulated annealing action optimization.

First, we implemented the direct policy algorithm of Peshkin et al. (2000). This algorithm is designed to learn policies for MDPs on factored state and action spaces. To parameterize the policy, we used a feed-forward neural network with one hidden layer. The number of hidden units was chosen to match the number of hidden variables in the competing restricted Boltzmann machine. The output layer of the neural network consisted of a softmax unit for each action variable, which gave the probability of executing each value for that action variable. For example, if an action variable has four possible values, then there are separate inputs (weights and activations) entering the output unit for each of the four possible values. The output unit produces four normalized probabilities by first exponentiating the values, and then normalizing by the sum of the four exponentiated values.

The parameterized policy is therefore factored. In other words, each action variable in the collective action is selected independently, given the probabilities expressed by the softmax output units. However, the hidden layer of the network allows the policy to ”co-ordinate” these probabilities conditioned on the state.

Second, we implemented an action-value function approximation using a feed-forward neural network with one hidden layer (Bertsekas and Tsitsiklis, 1996). The output of this network was a linear unit which gave the estimated value for the state and action presented on the input units. We used the SARSA algorithm to modify the parameters of the network (Eq.5). The gradient was computed using error backpropagation (Rumelhart et al., 1986). We used a greedy-epsilon exploration strategy, where the optimal action was approximated by simulated annealing. The number of iterations of simulated annealing was matched to the number of iterations of sampling used to select actions from the restricted Boltzmann machine, and the number of hidden units was matched to the number of hidden variables in the corresponding restricted Boltzmann machine.

7.1 The Large-Action Task

This task is designed to test value representation and action selection in a large action space, and is not designed to test temporal credit assignment. The large-action task has only immediate rewards. The state at each point in time is selected independent of previous states. The update rules were therefore used with the discount factor γ set to zero.

Consider a version of the task with an N-bit action. The task is generated as follows: Some small number of state-action pairs are randomly selected. We will call these “key” pairs. During execution, a state is chosen at random, and an action is selected by the learner. A reward is then generated by finding the key state closest to the current state (in Hamming distance). The reward received by the learner is equal to the number of bits that match between the key-action for this key state and the current action. So if the agent selects the key action it receives the maximum reward of N. The reward for any other action is found by subtracting the number of incorrect bits from N. The task (for $N = 5$) is illustrated in Figure 3.

A restricted Boltzmann machine with 13 hidden variables was trained on an instantiation of the large action task with an 12-bit state space and a 40-bit action space. Thirteen key states were

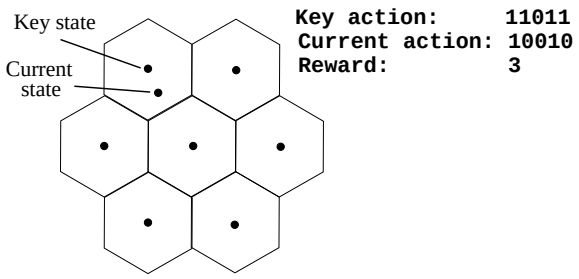


Figure 3: The large action task. The space of state bit vectors is divided into clusters of those which are nearest to each "key" state. Each key state is associated with a key action. The reward received by the learner is the number of bits shared by the selected action and the key action for the current state.

randomly selected. The network was run for 12 000 actions with a learning rate going from 0.1 to 0.01 and temperature going from 1.0 to 0.1 exponentially over the course of training. Each iteration was initialized with a random state. Each action selection consisted of 100 iterations of Gibbs sampling. The task was repeated 10 times for each method. The competing methods also had learning rates and (in the case of the backprop network) exploration schedules. The backprop network used a learning rate going from 0.005 to 0.004, and an ε-greedy exploration strategy going from 1 to 0 linearly over the course of the task. The direct policy method used a learning rate going from 0.1 to 0.01 over the course of the task. All learning parameters were selected by trial and error during preliminary experiments, with the best-performing parameters reported here.

Because the optimal action is known for each state we can compare the results to the optimal policy. We also compare to the two competing methods: the direct-policy method of Peshkin et al. (2000), and the feedforward neural network. The results are shown in Figure 4.

The learner must overcome two difficulties. First, it must find actions that receive rewards for a given state. Then, it must cluster the states which share commonly rewarded actions to infer the underlying key states. As the state space contains 2^{12} entries and the action space contains 2^{40} entries, this is not a trivial task. Yet the PoE achieves almost perfect performance after 12 000 actions. In comparison, the two other algorithms achieve suboptimal performance. The direct policy method seems to be particularly susceptible to local optima, yielding a large variance in solution quality. The backpropagation network may have continued to improve, given more training time.

7.2 The Blockers Task

The blockers task is a co-operative multi-agent task in which there are offensive players trying to reach an end zone, and defensive players trying to block them (see Figure 5).

The task is co-operative: As long as one agent reaches the end-zone, the "team" is rewarded. The team receives a reward of $+1$ when an agent reaches the end-zone, and a reward of -1 otherwise. The blockers are pre-programmed with a fixed blocking strategy. Each agent occupies one square on the grid, and each blocker occupies three horizontally adjacent squares. An agent cannot move into a square occupied by a blocker or another agent. The task has non-wrap-around edge conditions on the bottom, left and right sides of the field, and the blockers and agents can move up, down, left or right. Agents are ordered. If two agents want to move in to the same square, the first agent in

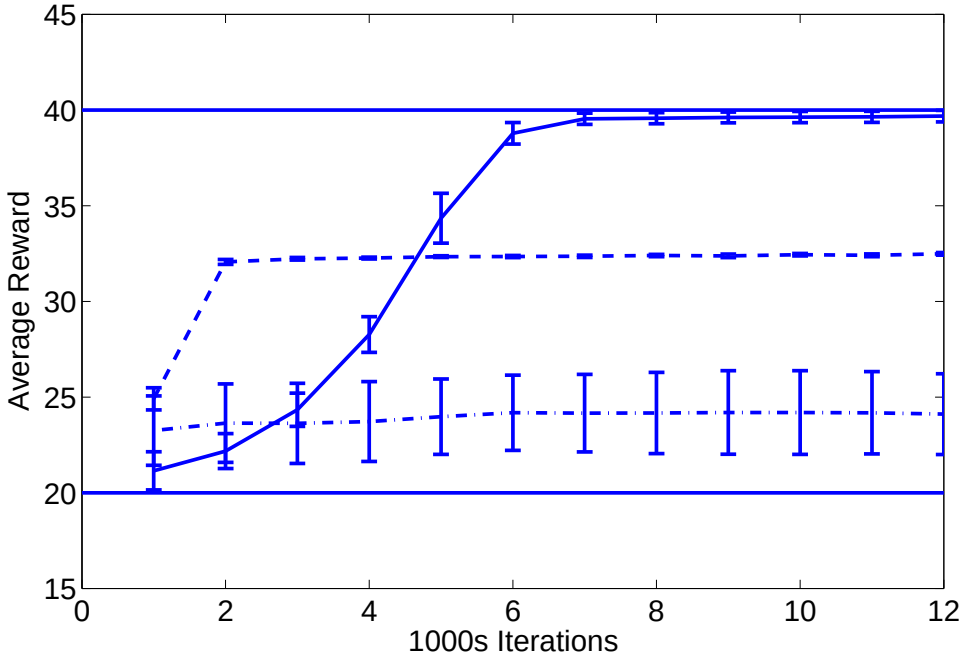


Figure 4: Results for the large action task. The graphs shows average reward versus iteration of training for three algorithms. The optimal policy gives an average reward of 40 (upper line). A random policy gives gives an average return of 20 (lower line). The solid line shows the PoE network, the dashed line shows the backprop network performance, and the dash-dotted line shows the direct policy method. Errorbars indicate 95% confidence intervals, computed across 10 repetitions of the task.

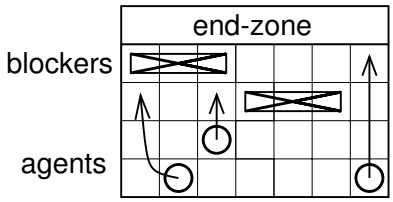


Figure 5: An example of the "blocker" task. Agents must get past the blockers to the end-zone. The blockers are pre-programmed with a strategy to stop them, but if the agents co-operate the blockers cannot stop them all simultaneously.

the ordering will succeed, and any others trying to move into that square will be unsuccessful. Note that later agents can not see the moves of earlier agents when making decisions. The ordering is just used to resolve collisions. If a move is unsuccessful, then the agent remains in its current square.

The blockers' moves are also ordered, but subsequent blockers do make decisions based on the moves of earlier blockers. The blockers operate a zone-based defense. Each blocker takes responsibility for a group of four columns. For example, blocker 1 is responsible for columns 1

through 4, blocker 2 is responsible for columns 4 through 7, and so on. If an agent moves into one of its columns and is in front of the end-zone, a blocker will move to block it. Because of the ordering, blockers will not move to stop agents that have already been stopped by other blockers.

A restricted Boltzmann machine with 4 hidden variables was trained using the SARSA learning rule on a 5×4 blocker task with two agents and one blocker. The collective state consisted of three position variables (two agents and one blocker) which could take on integer values $\{1,...,20\}$. The collective action consisted of two action variables taking on values from $\{1,...,4\}$. The PoE was compared to the backpropagation network and the direct policy method.

Each test was replicated 10 times. Each test run lasted for 300 000 collective actions, with a learning rate going from 0.1 to 0.01 linearly and temperature going from 1.0 to 0.01 exponentially over the course of training. Gibbs sampling and simulated annealing lasted for 10 iterations. The learning rates of the competing methods were the same as for the PoE network. The backpropagation network used an ε-greedy policy going linearly from 1 to 0 over the course of the task. The parameters for all of the methods were selected by trial and error using initial experiments, and the best performing values are reported here.

Each trial was terminated after either the end-zone was reached, or 20 collective actions were taken, whichever occurred first. Each trial was initialized with the blocker placed randomly in the top row and the agents placed randomly in the bottom row. The results are shown in Figure 6.

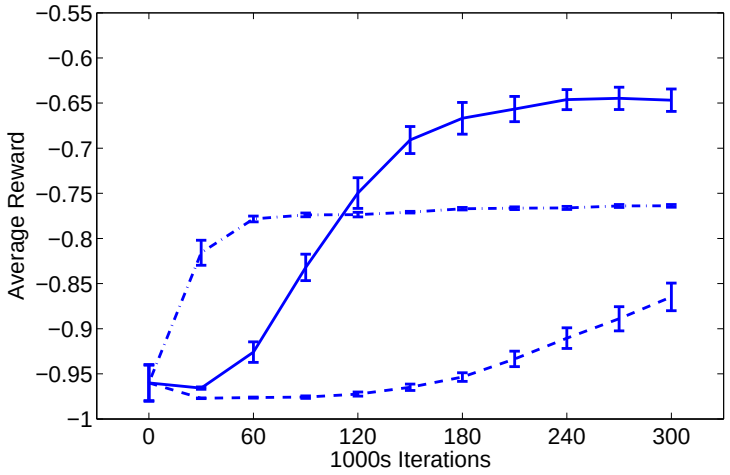


Figure 6: Results for the 2-agent blocker task. The graph shows average reward versus iteration of training for three algorithms. The solid line shows the PoE approximation; the dot-dashed line shows the direct policy method; and the dashed line shows the backprop network. The error bars show 95% confidence intervals.

Overall, the PoE network performs better than the two other algorithms. All three algorithms have the potential to find suboptimal local optima. Again, the direct policy algorithm seems to be particularly susceptible to this. The backprop network might have done better if it was allowed to continue training. The direct policy method finds a solution noticeably faster than the other two algorithms.

A restricted Boltzmann machine with 16 hidden variables was trained on a 4×7 blockers task with three agents and two blockers. Again, the input consisted of position variables for each blocker

and agent, and action variables for each agent. The network was trained for 300 000 collective actions, with a learning rate going from 0.1 to 0.05 linearly and temperature from 1 to 0.06 exponentially over the course of the task. Each trial was terminated after either the end-zone was reached, or 40 steps were taken, whichever occurred first. Again, the two competitor algorithms were also used. Each competitor had 16 hidden units, and simulated annealing and Gibbs sampling lasted for 10 iterations. The competing methods used the same learning rates and exploration strategy as in the previous experiment. Again, the task was replicated 10 times for each algorithm. The results are shown in Figure 7.

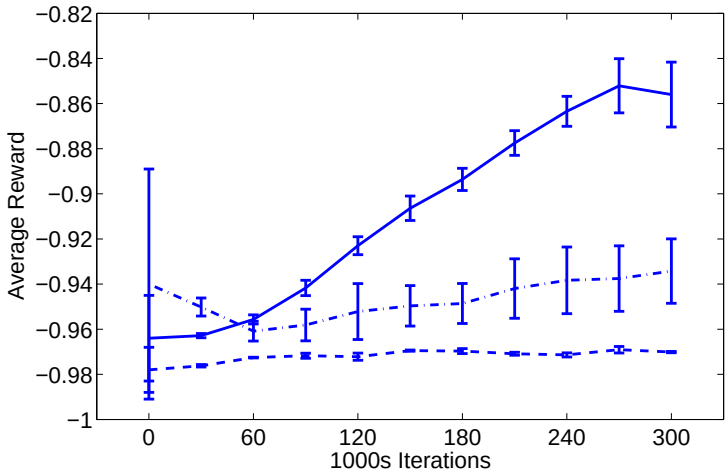


Figure 7: Results for the 3-agent blocker task. The graph shows average reward versus iteration of training for three algorithms. The solid line shows the PoE approximation; the dot-dashed line shows the direct policy method; and the dashed line shows the backprop network. The error bars show 95% confidence intervals.

In this larger version of the task, the backprop network does extremely poorly. The direct policy method does significantly worse than the PoE method. Of the three methods, the PoE method was able to find the best solution, although a suboptimal one. An example of a typical run for the 4×7 task is shown in Figure 8. The strategy discovered by the learner is to force the blockers apart with two agents, and move up the middle with the third. In the example, notice that Agent 1 seems to distract the "wrong" blocker given its earlier position. The agents in this example have learned a sub-optimal policy, where Agent 1 moves up as far as possible, and then left as far as possible, irrespective of its initial position.

Examples of features learned by the experts are shown in Figure 9. The hidden variables become active for a specific configuration in state space, and recommend a specific set of actions. Histograms below each feature indicate when that feature tends to be active during a trial. The histograms show that feature activity is localized in time. Features can be thought of as macro-actions or short-term policy segments. Each hidden variable becomes active during a particular "phase" of a trial, recommends the actions appropriate to that phase, and then ceases to be active.

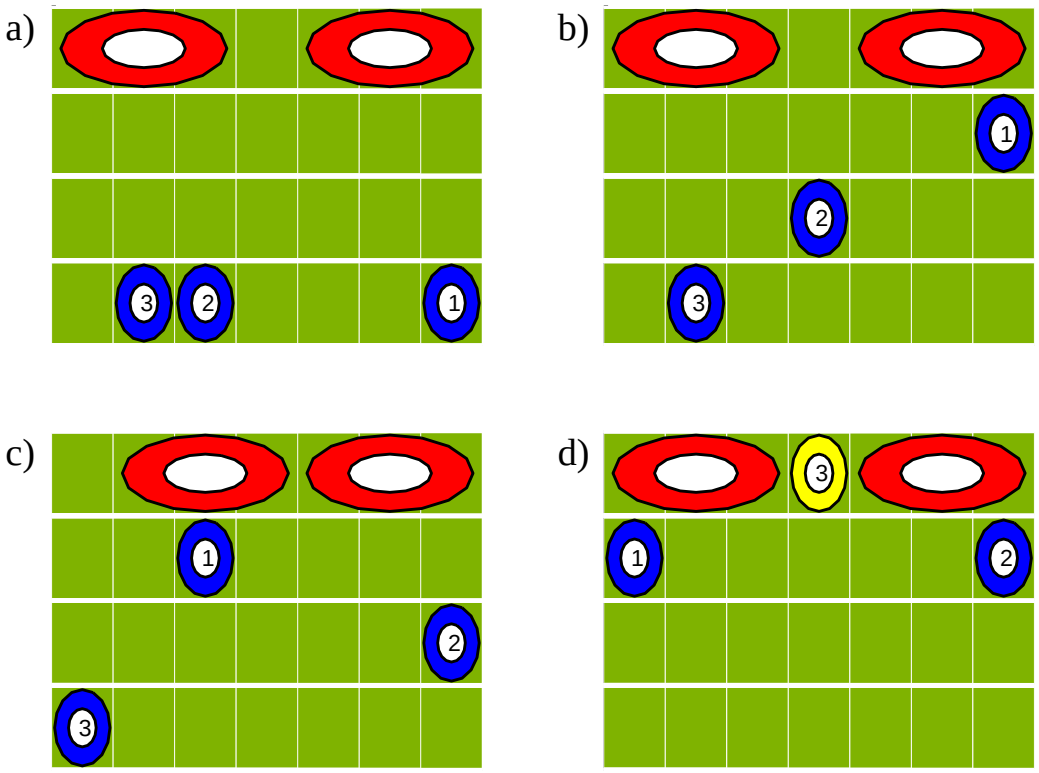


Figure 8: Example agent strategy after learning the 4×7 blocker task. a) The three agents are initialized to random locations along the bottom of the field. b) Two of the agents run to the top of the playing field. c) These two agents split and run to the sides. d) The third agent moves up the middle to the end-zone.

8. Discussion

The action sampling method is closely related to actor-critic methods (Sutton, 1984; Barto et al., 1983). An actor-critic method can be viewed as a biased scheme for selecting actions according to the value assigned to them by the critic. The selection is biased by the choice of actor parameterization. The sampling method of action selection is unbiased if the Markov chain is allowed to converge, but requires more computation. This is exactly the trade-off explored in the graphical models literature between the use of Monte Carlo inference (Neal, 1992) and variational approximations (Neal and Hinton, 1998; Jaakkola, 1997). Further, the resultant policy can potentially be more complicated than a typical parameterized actor would allow. This is because a parameterized distribution over actions has to be explicitly normalized. For example, an actor network might parameterize all policies in which the probability over each action variable is independent. This is the restriction implemented by Peshkin et al. (2000), and is also used for the direct policy method in our experimental section.

The sampling algorithm is also related to probability matching (Sabes and Jordan, 1996), in which good actions are made more probable under a model, and the temperature at which the prob-

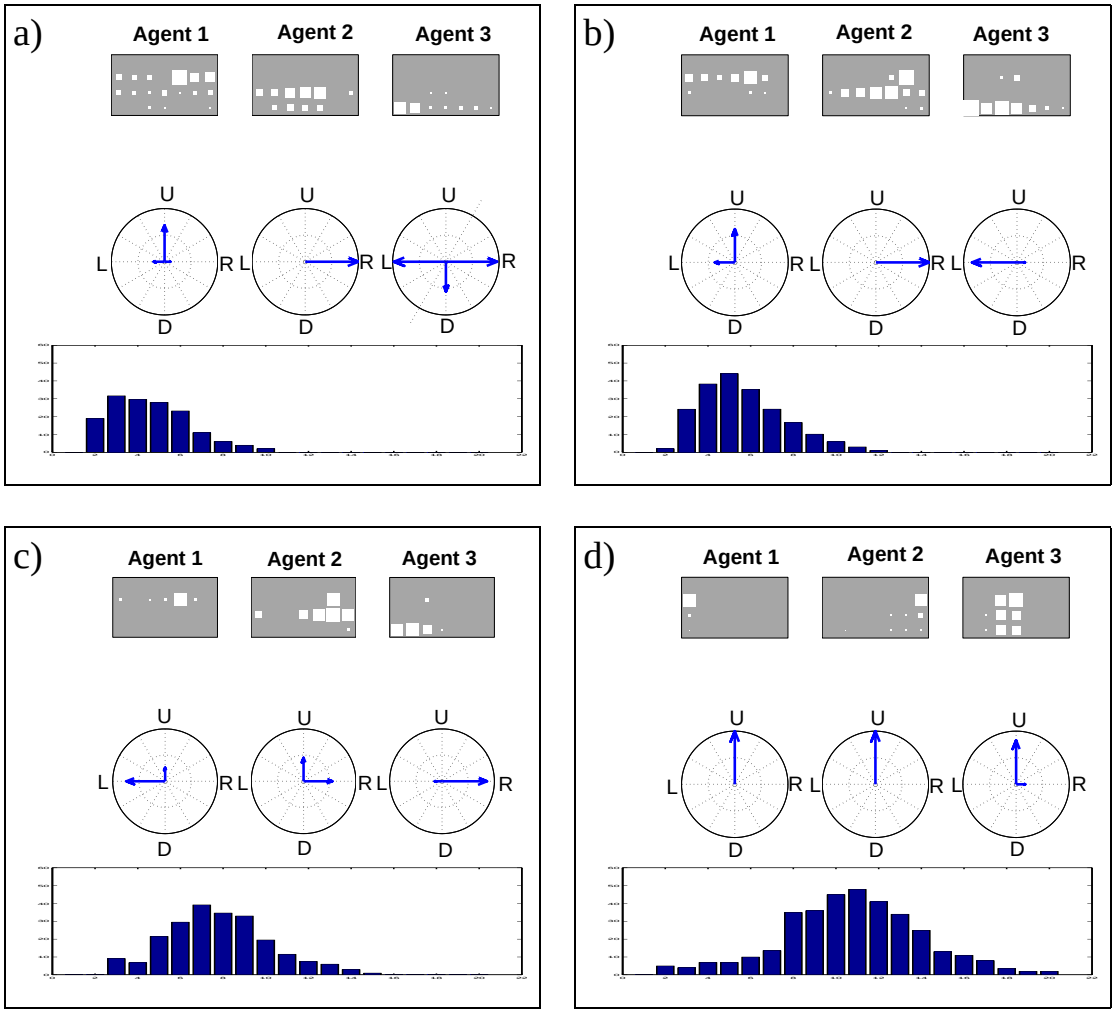


Figure 9: Features of the learned value function approximator for the 3-agent blocker task. The four features (a,b,c and d) correspond to the four stages shown in Figure 8. Each feature corresponds to a hidden variable in the RBM. The Hinton diagram shows where each of the three agents must be in order to "activate" the hidden variable (cause it to have a value of unity with high probability). The vector diagram indicates what actions are recommended by the hidden variable. The histogram is a plot of frequency of activation of the hidden variable versus time in a trial. It shows when during a run this feature tends to be active. The learned features are localized in state space and action space. Feature activity is localized in time.

ability is computed is slowly reduced over time in order to move from exploration to exploitation and avoid local minima. Unlike the sampling algorithm, the probability matching algorithm used a parameterized distribution which was maximized using gradient descent, and it did not address temporal credit assignment.

The PoE approximator could also be used in direct policy method. The network would directly encode the probabilities of selecting actions given states, rather than encoding the values of states and actions. Given a state, an action could be found using a sampling method. The sampling method would select actions approximately according to the policy encoded by the PoE.

Direct policy methods can be advantageous, because encoding the relative goodness of actions or a ranking of actions might be simpler than encoding actual values. That is, the optimal policy might be easier to learn than the value function. A PoE could be used with any direct policy method that only requires samples from the policy. This is because it is in general intractable to evaluate the probabilities of actions encoded in the PoE network, but possible to approximately sample actions using an MCMC method.

It is possible that the Gibbs sampling method which we use might not work well for some problems. In this case, other sampling methods could be used, which are better suited to avoiding local minima. While the need to sample actions using MCMC can be viewed as a disadvantage of our technique, an advantage is that improvements in sampling methods can be easily incorporated as they are developed.

8.1 Macro Actions

One way to interpret the individual experts in the product model is that they are learning "macro" or "basis" actions. As we have seen with the Blockers task, the hidden variables come to represent sets of actions that are spatially and temporally localized. We can think of the hidden variables as representing "basis" actions that can be combined to form a wide array of possible actions. The benefit of having basis actions is that it reduces the number of possible actions, thus making exploration more efficient. The drawback is that if the set of basis actions do not span the space of all possible actions, some actions become impossible to execute. By optimizing the set of basis actions during reinforcement learning, we find a set that can form useful actions, while excluding action combinations that are either not seen or not useful.

The "macro" actions learned by the PoE should not be confused with "temporally abstract actions". The learning and use of temporally abstract actions is an important area of current research in reinforcement learning (Parr and Russell, 1998; Precup et al., 1998; McGovern, 1998; Dietterich, 2000). The "macro" actions learned by the PoE have some features in common with these temporally abstract actions. In particular, the PoE macro actions tend to remain active for temporally prolonged periods. However, that does not make them temporally abstract actions. They do not come with the formal machinery of most temporally abstract actions (such as termination conditions), and it would be difficult to fit them in to one of the existing frameworks for temporal abstraction. The PoE "basis" actions should be thought of as finding a smaller subset of "good" actions within a large space of possible actions.

This suggests one way to improve the performance of an existing PoE solution. If the solution is performing poorly, it could be because some of the action space is not spanned by basis actions. Adding and learning parameters for additional hidden variables, while holding the parameters of the pre-existing variables constant, would allow the policy to improve without having to re-learn the entire solution. Similarly, if some useful collective actions are known a priori, they can be "hard-coded" into the PoE by fixing the hidden-action weights, and allowing the PoE to learn when (in which collective states) to use them.

8.2 Conditional Completion

If the action allowed at a particular time step is constrained, it is natural to want to know what is the best action consistent with the constraints. For example, if one of the agents in the blocker task becomes unable to move in directions other than up, we would like to ask for actions for the other agents that are consistent with this restriction. Sampling allows us to do this easily by fixing a subset of action variables to their required values, and sampling the rest. The result is a set of good values for some action variables conditioned on the fixed values of the others.

Similarly, we can fix only some of the state variables, and sample others. No doubt this would be most useful for data completion: If a state variable is missing, it would be nice to fill it in with its most probable value, conditioned on the others. Unfortunately we can not do this with a single PoE model. Instead of filling in values according to how probable they are under the dynamics of the environment, it will fill in values that yield high expected returns. In other words, the values that will be filled in for state variables will be those that are most desirable, not most probable. This could be used for an optimistic form of state completion, giving an upper bound on what reward we expect to see given that we do not really know what values those state variables take on. This could also be used to identify valuable "target" states that should be achieved if possible.

9. Summary

In this article we have shown that a combination of probabilistic inference, model learning and value function approximation allows for the solution of large Markov decision processes with factored states and actions. We have shown that the sampling technique can select actions in large action spaces (40 bit actions). We have drawn links between approximate inference, state representation and action selection. Future research on hierarchical value functions and directly learning stochastic policies represented as PoE models might be particularly fruitful.

Acknowledgments

The authors would like to thank Radford Neal, Craig Boutilier, Zoubin Ghahramani, Peter Dayan, Mike Mozer, Yee-Whye Teh and Andy Brown for valuable discussions and suggestions, and the anonymous referees for many helpful comments which substantially improved the paper. The majority of this research was completed while the authors were at the University of Toronto and the Gatsby Computational Neuroscience Unit, University College London. This research was supported by the Natural Science and Engineering Research Council of Canada and the Gastby Charitable Foundation. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture and by the Austrian Federal Ministry for Transport, Innovation and Technology.

Appendix A.

In this appendix we give some derivations related to restricted Boltzmann machines. First, we show that, for a restricted Boltzmann machine, the posterior distribution over hidden variables given visible variables factors into the product of the posterior distributions over each individual hidden variable. Second, we show that the two expressions for the equilibrium free energy are equivalent,

and compute the derivative of the equilibrium free energy of a restricted Boltzmann machine with respect to a parameter.

Given a set of binary random variables $\mathbf{V}$ with values $\mathbf{v}$, binary hidden variables $\mathbf{H}$ with values $\mathbf{h}$, and symmetric weighted edge w_{ik} connecting visible variable i to hidden variable k, the equilibrium free energy is given by

$$F(\mathbf{v}) = \langle E(\mathbf{v},\mathbf{h})\rangle_{P(\mathbf{h}|\mathbf{v})} + \langle \log P(\mathbf{h}|\mathbf{v})\rangle_{P(\mathbf{h}|\mathbf{v})}.$$

In the above, $E(\mathbf{v},\mathbf{h})$ denotes the energy, $P(\mathbf{h}|\mathbf{v})$ denotes the posterior distribution of the hidden variables given the visible variables, and $\langle\cdot\rangle_P$ denotes an expectation with respect to distribution P (see Section 5.1).

Consider the posterior distribution over the hidden variables. In the following, v_i denotes the value of visible variable i, and h_k denotes the value of hidden variable k. The notation $\sum_{\widehat{\mathbf{h}}}$ denotes a summation over all possible assignments to the binary variables in the set $\mathbf{H}$.

$$\begin{aligned}
P(\mathbf{h}|\mathbf{v}) &= \frac{\exp\{\sum_{i,k} w_{ik} v_i h_k\}}{\sum_{\widehat{\mathbf{h}}} \exp\{\sum_{i,k} w_{ik} v_i \widehat{h}_k\}} \\
&= \frac{\prod_k \exp\{\sum_i w_{ik} v_i h_k\}}{\sum_{\widehat{\mathbf{h}}} \prod_k \exp\{\sum_i w_{ik} v_i \widehat{h}_k\}} \\
&= \frac{\prod_k \exp\{\sum_i w_{ik} v_i h_k\}}{\prod_k \sum_{\widehat{h}_k=0}^{1} \exp\{\sum_i w_{ik} v_i \widehat{h}_k\}} \\
&= \prod_k \frac{\exp\{\sum_i w_{ik} v_i h_k\}}{\sum_{\widehat{h}_k=0}^{1} \exp\{\sum_i w_{ik} v_i \widehat{h}_k\}} \\
&= \prod_k P(h_k|\mathbf{v}).
\end{aligned}$$

The posterior factors into the product of the posterior distributions over each separate hidden variable, given the values of the visible variables. The posterior over hidden variables can be computed efficiently, because each individual hidden-unit posterior is tractable:

$$P(h_k = 1|\mathbf{v}) = \sigma\left(E(\mathbf{v}, h_k = 1)\right),$$

where $\sigma(\cdot)$ denotes the logistic function: $\sigma(x) = 1/(1+e^{-x})$.

We will now compute the derivative of the equilibrium free energy with respect to a weight. We follow the technique of (Hertz et al., 1991). First, we prove the correspondence between the negative equilibrium free energy and the log of the normalizing constant of the posterior distribution (see Eqs. 6 and 9) :

$$\begin{aligned}
F(\mathbf{v}) &= \langle E(\mathbf{v},\mathbf{h})\rangle_{P(\mathbf{h}|\mathbf{v})} + \langle \log P(\mathbf{h}|\mathbf{v})\rangle_{P(\mathbf{h}|\mathbf{v})} \\
&= \langle E(\mathbf{v},\mathbf{h})\rangle_{P(\mathbf{h}|\mathbf{v})} + \langle \log P(\mathbf{h}|\mathbf{v})\rangle_{P(\mathbf{h}|\mathbf{v})} \\
&\qquad + \log \sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\} - \log \sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\} \\
&= -\langle \log \exp\{-E(\mathbf{v},\mathbf{h})\}\rangle_{P(\mathbf{h}|\mathbf{v})} + \langle \log P(\mathbf{h}|\mathbf{v})\rangle_{P(\mathbf{h}|\mathbf{v})} \\
&\qquad + \log \sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\} - \log \sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\}
\end{aligned}$$

$$
\begin{aligned}
&= -\left\langle \log \frac{\exp\{-E(\mathbf{v},\mathbf{h})\}}{\sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\}} \right\rangle_{P(\mathbf{h}|\mathbf{v})} \\
&\qquad + \langle \log P(\mathbf{h}|\mathbf{v}) \rangle_{P(\mathbf{h}|\mathbf{v})} - \log \sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\} \\
&= -\langle \log P(\mathbf{h}|\mathbf{v}) \rangle_{P(\mathbf{h}|\mathbf{v})} + \langle \log P(\mathbf{h}|\mathbf{v}) \rangle_{P(\mathbf{h}|\mathbf{v})} - \log \sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\} \\
&= -\log \sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\} \\
&= -\log Z_h, \qquad (11)
\end{aligned}
$$

where Z_h denotes the normalizing constant of the posterior distribution over the hidden variables given the visible variables.

Next, we can take the derivative of $-\log Z_h$ with respect to a weight w_{ik}.

$$
\begin{aligned}
-\frac{\partial \log Z_h}{\partial w_{ik}} &= -\frac{1}{Z_h}\frac{\partial Z_h}{\partial w_{ik}} \\
&= \frac{-1}{Z_h}\frac{\partial}{\partial w_{ik}}\left[\sum_{\widehat{\mathbf{h}}} \exp\{-E(\mathbf{v},\widehat{\mathbf{h}})\}\right] \\
&= \frac{-1}{Z_h}\frac{\partial}{\partial w_{ik}}\left[\sum_{\widehat{\mathbf{h}}} \exp\{\sum_{i,k} w_{ik} v_i \widehat{h}_k\}\right] \\
&= \sum_{\widehat{\mathbf{h}}} \frac{-1}{Z_h}\left[v_i \widehat{h}_k \exp\{\sum_{i,k} w_{ik} v_i \widehat{h}_k\}\right] \\
&= -\sum_{\widehat{\mathbf{h}}} \frac{\exp\{\sum_{i,k} w_{ik} v_i \widehat{h}_k\}}{Z_h} v_i \widehat{h}_k \\
&= -\sum_{\widehat{\mathbf{h}}} P(\widehat{\mathbf{h}}|\mathbf{v}) v_i \widehat{h}_k \\
&= -v_i \langle h_k \rangle_{P(\mathbf{h}|\mathbf{v})}.
\end{aligned}
$$

Thus, the derivative of the equilibrium free energy with respect to a weight is simply the expected value of the hidden variable, times the value of the visible variable.

References

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

L. Baird and H. Klopf. Reinforcement learning with high-dimensional continuous actions. Technical Report WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air Force Base, 1993.

L. Baird and A. Moore. Gradient descent for general reinforcement learning. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. The MIT Press, Cambridge, 1999.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, 13:835–846, 1983.

R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957a.

R. E. Bellman. A Markov decision process. *Journal of Mathematical Mechanics*, 6:679–684, 1957b.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

C. Boutilier, F. Bacchus, and R. I. Brafman. Ucp-networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 56–64, 2001.

C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*, pages 71–80, 1999.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Articial Intelligence*, 121:49–107, 2000.

C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proc. AAAI-96*, 1996.

A. D. Brown and G. E. Hinton. Products of hidden Markov models. In T. S. Jaakkola and T. Richardson, editors, *Proceedings of Artificial Intelligence and Statistics 2001*, pages 3–11, 2001.

G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999.

T. Dean, R. Givan, and K. Kim. Solving stochastic planning problems with large state and action spaces. In *Proc. Fourth International Conference on Artificial Intelligence Planning Systems*, 1998.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5, 1989.

T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195: 216–222, 1987.

Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. In John E. Moody, Steven J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan Kaufmann , San Mateo, 1992.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems*, volume 14. The MIT Press, Cambridge, 2002.

J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA, 1991.

G. E. Hinton. Products of experts. In *ICANN-99*, volume 1, pages 1–6, 1999.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14 (8):1771–1800, 2002.

G. E. Hinton and T. J. Sejnowski. *Parallel Distributed Processing*, volume 1, chapter Learning and Relearning in Boltzman Machines, pages 282–317. The MIT Press, Cambridge, 1986.

R. A. Howard. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, MA, 1960.

T. S. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models*. Department of Brain and Cognitive Sciences, MIT, Cambridge, MA, 1997. Ph.D. thesis.

T. S. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 345–352. The MIT Press, Cambridge, 1995.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183:233, 1999.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Series D, Journal of Basis Engineering*, 82:35–45, March 1960.

W. S. Lovejoy. A survey of algorithmic methods for partially observable Markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

D. A. McAllester and S. P. Singh. Approximate planning for factored POMDPs using belief state simplification. In *Proc. UAI'99*, 1999.

A. McGovern. acQuire-macros: An algorithm for automatically learning macro-actions. In *NIPS98 Workshop on Abstraction and Hierarchy in Reinforcement Learning*, 1998.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

N. Meuleau, M. Hauskrecht, K-E Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *Proc. AAAI-98*, 1998.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

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 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. Kluwer Academic Publishers, 1998.

R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, Cambridge, 1998.

L. Peshkin, K-E Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

L. Peshkin, N. Meuleau, and L. P. Kaelbling. Learning policies with external memory. In *Proc. 16th International Conference on Machine Learning*, pages 307–314, 1999.

R. B. Potts. Some generalized order-disorder transformations. *Proc. Camb. Phil. Soc.*, 48:106–109, 1952.

P. Poupart and C. Boutilier. Vector-space analysis of belief-state approximation for POMDPs. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, 2001.

D. Precup, R. Sutton, and S. Singh. Theoretical results on reinforcement learning with temporally abstract options. In *Proceedings of the Tenth European Conference on Machine Learning*, 1998.

L. R. Rabiner and B.-H. Juang. An introduction to hidden Markov models. *IEEE ASSAP Magazine*, 3:4–16, January 1986.

A. Rodriguez, R. Parr, and D. Koller. Reinforcement learning using approximate belief states. In S. A. Solla, T. K. Leen, and K-R Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. The MIT Press, Cambridge, 2000.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by back-propagating errors. *Nature*, 323:533–536, 1986.

G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, 1994.

P. N. Sabes and M. I. Jordan. Reinforcement learning by probability matching. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1080–1086. The MIT Press, Cambridge, 1996.

B. Sallans. Learning factored representations for partially observable Markov decision processes. In S. A. Solla, T. K. Leen, and K-R Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 1050–1056. The MIT Press, Cambridge, 2000.

J. C. Santamaria, R. S. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2), 1998.

S. Singh, T. Jaakkola, M. Littmann, and C. Szepasvfiri. Convergence results for single-step on-policy reinforcementlearning algorithms. *Machine Learning*, 38(3):287–308, 2000.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. The MIT Press, Cambridge, MA, 1986.

R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. University of Massachusetts, Amherst, 1984. Ph.D. thesis.

R. S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1038–1044. The MIT Press, Cambridge, 1996.

R. S Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

S. Thrun. Monte Carlo POMDPs. In S. A. Solla, T. K. Leen, and K-R Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. The MIT Press, Cambridge, 2000.

Journal of Machine Learning Research 5 (2004) 1089–1105 Submitted 05/03; Revised 9/03; Published 9/04

No Unbiased Estimator of the Variance of K-Fold Cross-Validation

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Dept. IRO, Université de Montréal
C.P. 6128, Montreal, Qc, H3C 3J7, Canada

Yves Grandvalet YVES.GRANDVALET@UTC.FR
Heudiasyc, UMR CNRS 6599
Université de Technologie de Compiègne, France

Editor: Dana Ron

Abstract

Most machine learning researchers perform quantitative experiments to estimate generalization error and compare the performance of different algorithms (in particular, their proposed algorithm). In order to be able to draw statistically convincing conclusions, it is important to estimate the uncertainty of such estimates. This paper studies the very commonly used K-fold cross-validation estimator of generalization performance. The main theorem shows that there exists no universal (valid under all distributions) unbiased estimator of the variance of K-fold cross-validation. The analysis that accompanies this result is based on the eigen-decomposition of the covariance matrix of errors, which has only three different eigenvalues corresponding to three degrees of freedom of the matrix and three components of the total variance. This analysis helps to better understand the nature of the problem and how it can make naive estimators (that don't take into account the error correlations due to the overlap between training and test sets) grossly underestimate variance. This is confirmed by numerical experiments in which the three components of the variance are compared when the difficulty of the learning problem and the number of folds are varied.

Keywords: cross-validation, variance estimators, k-fold cross-validation, statistical comparisons of algorithms

1. Introduction

In machine learning, the standard measure of accuracy for trained models is the prediction error (PE), i.e. the expected loss on future examples. Learning algorithms themselves are often compared according to their average performance, which is formally defined as the expected value of prediction error (EPE) over training sets.

When the data distribution is unknown, PE and EPE cannot be computed. If the amount of data is large enough, PE can be estimated by the mean error over a hold-out test set. The usual variance estimates for means of independent samples can then be computed to derive error bars on the estimated prediction error, and to assess the statistical significance of differences between models.

The hold-out technique does not account for the variance with respect to the training set, and may thus be considered inappropriate for the purpose of algorithm comparison (Dietterich, 1999). Moreover, it makes an inefficient use of data which forbids its application to small sample sizes. In

this situation, one rather uses computer intensive resampling methods such as cross-validation or bootstrap to estimate PE or EPE.

We focus here on K-fold cross-validation. While it is known that cross-validation provides an unbiased estimate of EPE, it is also known that its variance may be very large (Breiman, 1996). This variance should be estimated to provide faithful confidence intervals on PE or EPE, and to test the significance of observed differences between algorithms. This paper provides theoretical arguments showing the difficulty of this estimation.

The difficulties of the variance estimation have already been addressed (Dietterich, 1999; Kohavi, 1995; Nadeau and Bengio, 2003). Some distribution-free bounds on the deviations of cross-validation are available, but they are specific to some locally defined decision rules, such as nearest neighbors (Devroye et al., 1996). This paper builds upon the work of Nadeau and Bengio (2003), which investigated in detail the theoretical and practical merits of several estimators of the variance of cross-validation. Our analysis departs from this work in the sampling procedure defining the cross-validation estimate. While Nadeau and Bengio (2003) consider K independent training and test splits, we focus on the standard K-fold cross-validation procedure, where there is no overlap between test sets: each example of the original data set is used once and only once as a test example.

This paper is organized as follows. Section 2 defines the measures of performance for algorithms, their estimation by K-fold cross-validation and similar procedures such as delete-m jackknife. Our theoretical findings are summarized in Sections 3–6. They are followed in Section 7 by experiments illustrating the effect of experimental conditions on the total variance and its decomposition in three components, and confirming the underestimation of variance obtained by the naive estimator commonly used by researchers.

2. General Framework

In machine learning, the performance measure differs according to the experimenter's viewpoint. In applications, we are interested in finding the best algorithm for solving the particular task at hand, specified by one particular training set and some information about the data generating process. In algorithm evaluation, we want to compare several learning algorithms for different learning tasks, and we care about the sensitivity of the learning algorithm to the choice of training examples.

2.1 Measures of Performance

Formally, we have a training set $D = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, with $\mathbf{z}_i \in \mathcal{Z}$, independently sampled from an unknown distribution P. We also have a learning algorithm A, which maps a data set of (almost) arbitrary size to a function $A : \mathcal{Z}^* \to \mathcal{F}$. Throughout this paper, we consider symmetric algorithms, i.e. A is insensitive to the ordering of examples in the training set D. The discrepancy between the prediction and the observation $\mathbf{z}$ is measured by a loss functional $L : \mathcal{F} \times \mathcal{Z} \to \mathbb{R}$. Typically, L is the quadratic loss in regression ($L(f,(x,y)) = (f(x)-y)^2$) and the misclassification $\{0,1\}$-loss in classification ($(L(f,(x,y)) = 1_{f(x)\neq y}$).

Let $f = A(D)$ be the function returned by algorithm A on the training set D. In application based evaluation, the goal of learning is usually stated as the minimization of the prediction error, i.e. the expected loss on future test examples

$$\mathrm{PE}(D) = E[L(f,\mathbf{z})], \tag{1}$$

where the expectation is taken with respect to $\mathbf{z}$ sampled from P.[1]

In algorithm based evaluation, we are not really interested in performances on a specific training set; we would like comparisons on a more general basis. In this context, the lowest level of generality can be stated as "training sets of size n sampled from P", and the performance of learning algorithm A can be measured by the expected performance of the functions returned in this situation

$$\mathrm{EPE}(n) = E[L(A(D), \mathbf{z})], \tag{2}$$

where the expectation is taken with respect to D sampled from P^n and $\mathbf{z}$ independently sampled from P.

Note that other types of performances measure can be proposed, based for example on parameters, or defined by the predictability in other frameworks, such as the prequential analysis (Dawid, 1997).

When the data distribution is unknown, PE and EPE cannot be computed. They have to be estimated, and it is often crucial to assess the uncertainty attached to this estimation:

- in application-oriented experiments, to give a confidence interval on PE;
- in algorithm-oriented experiments, to take into account the stability of a given algorithm. For comparisons between algorithms, it is essential to assess the statistical significance of observed differences in the estimate $\widehat{\mathrm{EPE}}$.

Although this point is often overlooked, estimating the variance of the estimates $\widehat{\mathrm{PE}}$ and $\widehat{\mathrm{EPE}}$ requires caution.

2.2 Hold-Out Estimates of Performance

If the amount of data is large enough, PE can be estimated by the mean error over a hold-out test set, and the usual variance estimate for means of independent variables can then be computed. However, even in the ideal situation where several independent training and test sets would be available, this estimate should not be applied to compute the variance of $\widehat{\mathrm{EPE}}$: even though training and test examples are independent, the test errors are correlated, since many test errors are computed for each training set, now considered as a random variable.

Figure 1 illustrates how crucial it is to take these correlations into account. The mean of two variance estimators is reported *vs.* the empirical variance of the hold-out estimate, in an ideal situation where 10 independent training and test sets are available. The variance of $\widehat{\mathrm{EPE}}(n)$ (estimated on 100,000 independent experiments) is displayed for reference by the dotted line. The average of $\widehat{\theta}_1$, the variance estimator ignoring correlations, shows that this estimate is highly biased, even for large sample sizes, whereas the variance estimator $\widehat{\theta}_2$, taking into account correlations, is unbiased. The details of this experiment are given below.

Experiment 1 *Ideal hold-out estimate of* EPE.

We have $K = 10$ independent training sets $D_1, \ldots, D_K$ of n independent examples $\mathbf{z}_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x_i} = (x_{i1}, \ldots, x_{id})'$ is a d-dimensional centered, unit covariance Gaussian variable ($d = 30$),

1. Note that we are using the same notation for random variables and their realization. The intended meaning will be specified when not clear from the context.

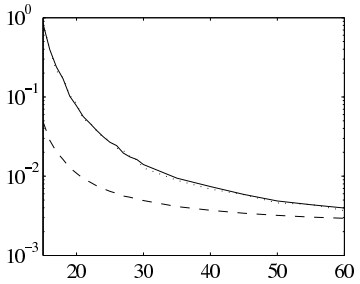


Figure 1: Estimates of the variance of $\widehat{\mathrm{EPE}}(n)$ vs. empirical variance of $\widehat{\mathrm{EPE}}(n)$ (shown by bold curve) on 100,000 experiments. The average of the variance estimators $\widehat{\theta}_1$ (ignoring correlations, dashed curve) and $\widehat{\theta}_2$ (taking into account correlations, dotted curve) are displayed for different training sample size n.

$y_i = \sqrt{3/d}\sum_{k=1}^{d} x_{ik} + \varepsilon_i$ with ε_i being independent, centered, unit variance Gaussian variables.[2] *We also have K independent test sets $T_1, \ldots, T_K$ of size n sampled from the same distribution.*

The learning algorithm consists in fitting a line by ordinary least squares, and the estimate of EPE *is the average quadratic loss on test examples $\widehat{\mathrm{EPE}} = \bar{L} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{n}\sum_{\mathbf{z}_i \in T_k} L_{ki}$, where $L_{ki} = L(A(D_k), \mathbf{z}_i)$.*

The first estimate of variance of $\widehat{\mathrm{EPE}}$ is $\widehat{\theta}_1 = \frac{1}{Kn(Kn-1)}\sum_{k=1}^{K}\sum_i (L_{ki} - \bar{L})^2$, which is unbiased provided there is no correlation between test errors. The second estimate is $\widehat{\theta}_2 = \frac{1}{K(K-1)n^2}\sum_{k=1}^{K}\sum_{i,j}(L_{ki} - \bar{L})(L_{kj} - \bar{L})$, which takes into account correlations between test errors.

Looking at Figure 1 suggests that asymptotically the naive estimator of variance converges to the true variance. This can be shown formally by taking advantage of the results in this paper, as long as the learning algorithm converges as the amount of training data goes to infinity (i.e. as $n \to \infty$ the function $A(D)$ obtained does not depend on the particular training set D). In that limit, the correlations between test errors converge to 0. The rate of convergence will depend on the stability of the learning algorithm as well as on the nature of the data distribution (e.g., the presence of thick tails and outliers will slow down convergence).

The hold-out technique makes an inefficient use of data which forbids its application in most real-life applications with small samples. Then, K-fold cross-validation can provide estimates of PE or EPE.

2.3 K-Fold Cross-Validation Estimates of Performance

Cross-validation is a computer intensive technique, using all available examples as training and test examples. It mimics the use of training and test sets by repeatedly training the algorithm K times with a fraction $1/K$ of training examples left out for testing purposes. This kind of hold-out estimate of performance lacks computational efficiency due to the repeated training, but the latter are meant to lower the variance of the estimate (Stone, 1974).

2. The $\sqrt{3/d}$ factor provides an R^2 of approximately $3/4$.

In practice, the data set D is first chunked into K disjoint subsets (or *blocks*) of the same size[3] $m \triangleq n/K$. Let us write T_k for the k-th such block, and D_k the training set obtained by removing the elements in T_k from D. The cross-validation estimator is defined as the average of the errors on test block T_k obtained when the training set is deprived from T_k:

$$\mathrm{CV}(D) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{m}\sum_{\mathbf{z}_i \in T_k} L(A(D_k), \mathbf{z}_i). \tag{3}$$

Does CV estimate PE or EPE? Such a question may seem pointless considering that $\mathrm{PE}(D)$ is an estimate of $\mathrm{EPE}(n)$, but it becomes relevant when considering the variance of CV: does it inform us of the uncertainty about PE or EPE?

On the one hand, only one training set, D, enters the definition of CV, which can be, up to an approximation, an unbiased estimate of $\mathrm{PE}(D)$ (Hastie and Tibshirani, 1990).[4] Some distribution-free bounds on the expected deviations of $|\mathrm{CV}(D) - PE(D)|$ are available for leave-one-out cross-validation applied to specific algorithms A such as nearest neighbors (Devroye et al., 1996). In a more general context, it has also been proved that, under suitable stability assumptions on the algorithm A, $\mathrm{CV}(D)$ estimates $\mathrm{PE}(D)$ at least as accurately as the training error (Kearns and Ron, 1996; Anthony and Holden, 1998). A more appealing result states that CV is a more accurate estimate of PE than hold-out testing (Blum et al., 1999). However, this statement does not apply to $\mathrm{PE}(D)$, but to the prediction error of a randomized algorithm picking solutions uniformly within $\{A(D_k)\}_{k=1}^{K}$.

On the other hand, CV is explicitly defined from the learning algorithm A, and not from the function $f = A(D)$. The inner average in the definition of CV (3) is an average test loss for $A(D_k)$ which thus estimates unbiasedly $\mathrm{PE}(D_k)$. The training sets $D_1, \ldots, D_K$ are clearly not independent, but they are sampled from P^{n-m}. Hence, the outer average of (3) estimates unbiasedly $\mathrm{EPE}(n-m)$.[5] Here, following Dietterich (1999) and Nadeau and Bengio (2003), we will adopt this latter point of view.

The variance estimate of $\widehat{\mathrm{EPE}}$ provided by the hold-out estimate has to account for test error dependencies due to the choice of training set, which cannot be estimated using a single training/test experiment. Here, the situation is more complex, since there are additional dependencies due to the overlapping training sets $D_1, \ldots, D_K$. Before describing this situation in detail and summarizing the results of our theoretical analysis in Sections 3–6, we detail some procedures similar to K-fold cross-validation, for which the forthcoming analysis will also hold.

2.4 Other Estimates of the K-Fold Cross-Validation Type

One of the main use of variance estimates of $\widehat{\mathrm{EPE}}$ is to compare learning algorithms. The analysis presented in this paper also applies to the version of cross-validation dedicated to this purpose: if we want to compare the performances of algorithms A_1 and A_2, cross-validation with matched pairs

3. To simplify the analysis below we assume that n is a multiple of K.
4. More precisely, following Hastie and Tibshirani (1990), when L is the quadratic loss, and writing $f = A(D)$, $f^{-k} = A(D_k)$, assuming that for $(\mathbf{x}_i, y_i) = \mathbf{z}_i \in T_k$, $\frac{1}{K}\sum_{k=1}^{K} f^{-k}(\mathbf{x}_i) \approx f(\mathbf{x}_i)$ (which is weaker than $f^{-k} \approx f$) yields $E[\mathrm{CV}] \approx E[\frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2]$, where the expectation is taken with respect to $y_1, \ldots, y_n$.
5. Note that leave-one-out cross-validation is known to fail to estimate EPE for unsmooth statistics (e.g. Breiman, 1996; Efron and Tibshirani, 1993). This failure is due to the similarity of the training sets $D_1, \ldots, D_K$ which are far from being representative samples drawn from P^{n-m}.

should be the method of choice

$$\Delta\mathrm{CV}(D) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{m}\sum_{\mathbf{z}_i\in T_k} L(A_1(D_k),\mathbf{z}_i) - L(A_2(D_k),\mathbf{z}_i). \tag{4}$$

Compared to the difference of two independent cross-validation estimates, ΔCV avoids the additional variability due to train/test splits.

In application oriented experiments, we would like to estimate $\mathrm{PE}(D)$, the expected error when training with the given D. We have seen in Section 2.3 that under stability assumptions, CV can be used to estimate PE. Alternatively, we may resort to the jackknife or the delete-m jackknife (see e.g. Efron and Tibshirani (1993)) to estimate the optimism (i.e. the bias of the mean error on training examples, when the latter is used to estimate $\mathrm{PE}(D)$). Ideally, the estimate of optimism should be an average over all subsets of size $n-m$, but a less computationally intensive alternative is

$$(K-1)\left(\frac{1}{K(n-m)}\sum_{k=1}^{K}\sum_{\mathbf{z}_i\in D_k} L(A(D_k),\mathbf{z}_i) - \frac{1}{n}\sum_{i=1}^{n} L(A(D),\mathbf{z}_i)\right). \tag{5}$$

The link with cross-validation is exhibited more clearly by the following expression of the (debiased) jackknife estimate of PE

$$\mathrm{JK} = \mathrm{CV} + \frac{1}{n}\sum_{k=1}^{K}\sum_{i=1}^{n}\left(L(A(D),\mathbf{z}_i) - L(A(D_k),\mathbf{z}_i)\right). \tag{6}$$

For additional information about jackknife estimates and clues on the derivation of (5) and (6), the reader is referred to Efron and Tibshirani (1993).

2.5 Generic Notations

This paper studies the variance of statistics such as CV, ΔCV or JK. In what follows, these statistics will be denoted by $\hat{\mu}$, a generic notation for means of observations e_i split in K groups.

$$\begin{aligned}\hat{\mu} &= \frac{1}{n}\sum_{i=1}^{n} e_i \\ &= \frac{1}{K}\sum_{k=1}^{K}\frac{1}{m}\sum_{i\in T_k} e_i,\end{aligned}$$

where, slightly abusing notation, $i\in T_k$ means $\mathbf{z}_i\in T_k$ and

$$\forall i\in T_k,\ e_i = \begin{cases} L(A(D_k),\mathbf{z}_i) & \text{for } \hat{\mu}=\mathrm{CV},\\ L(A_1(D_k),\mathbf{z}_i) - L(A_2(D_k),\mathbf{z}_i) & \text{for } \hat{\mu}=\Delta\mathrm{CV},\\ KL(A(D),\mathbf{z}_i) - \sum_{\ell\neq k} L(A(D_\ell),\mathbf{z}_i) & \text{for } \hat{\mu}=\mathrm{JK}.\end{cases}$$

Note that $\hat{\mu}$ is the average of identically distributed (dependent) variables. Thus, it asymptotically converges to a normally distributed variable, which is completely characterized by its expectation $E[\hat{\mu}]$ and its variance $\mathrm{Var}[\hat{\mu}] = E[\hat{\mu}^2] - E[\hat{\mu}]^2$.

3. Structure of the Covariance Matrix

The variance of $\hat{\mu}$ is defined as follows

$$\theta = \frac{1}{n^2} \sum_{i,j} \mathrm{Cov}(e_i, e_j),$$

where $\mathrm{Cov}(e_i, e_j) = E[e_i e_j] - E[e_i]E[e_j]$ is the covariance between variables e_i and e_j.

By using symmetry arguments over permutations of the examples in D, we show that many distributions on e_i and pairwise joint distributions on (e_i, e_j) are identical. As a result, the covariance matrix Σ has a very particular block structure, with only three possible values for $\Sigma_{ij} = \mathrm{Cov}(e_i, e_j)$, and the expression of θ is thus a linear combination of these three values.

Lemma 1 *Using the notation introduced in section 2.5,*

1. *all e_i are identically distributed:*
   there exists f such that, $\forall i,\ P(e_i = u) = f(u)$.
2. *all pairs (e_i, e_j) belonging to the same test block are jointly identically distributed:*
   there exists g such that, $\forall (i,j) \in T_k^2 : j \neq i,\ P(e_i = u, e_j = v) = g(u,v)$.
3. *all pairs (e_i, e_j) belonging to different test blocks are jointly identically distributed:*
   there exists h such that, $\forall i \in T_k,\ \forall j \in T_\ell : \ell \neq k,\ P(e_i = u, e_j = v) = h(u,v)$.

Proof
These results are derived immediately from the permutation-invariance of $P(D)$ and the symmetry of A.

- *invariance with respect to permutations within test blocks:*
  1. $\forall (i,i') \in T_k^2,\ P(e_i = u) = P(e_{i'} = u) = f_k(u)$;
     $\forall (i,i') \in T_k^2,\ \forall j \in T_\ell$:
     $P(e_i = u, e_j = v) = P(e_{i'} = u, e_j = v)$
     hence:
  2. $\forall (i,j) \in T_k^2 : j \neq i,\ P(e_i = u, e_j = v) = g_k(u,v)$.
  3. $\forall i \in T_k,\ \forall j \in T_\ell : \ell \neq k,\ P(e_i = u, e_j = v) = h_{k\ell}(u,v)$.
- *invariance with respect to permutations between test blocks.*
  1. $\forall (k,k'),\ f_k(u) = f_{k'}(u) = f(u)$;
  2. $\forall (k,k'),\ g_k(u,v) = g_{k'}(u,v) = g(u,v)$;
  3. $\forall (k,k'),\ \forall (\ell,\ell') : \ell \neq k, \ell \neq k', \ell' \neq k, \ell' \neq k',\ h_{k\ell}(u,v) = h_{k\ell'}(u,v) = h_{k'\ell'}(u,v) = h_{k'\ell}(u,v) = h(u,v)$.

Q.E.D.

Corollary 2 *The covariance matrix Σ of cross-validation errors $\mathbf{e} = (e_1, \ldots, e_n)'$ has the simple block structure depicted in Figure 2:*

1. *all diagonal elements are identical*

   $\forall i,\ \mathrm{Cov}(e_i, e_i) = \mathrm{Var}[e_i] = \sigma^2$;

2. *all the off-diagonal entries of the K $m \times m$ diagonal blocks are identical*

   $\forall (i,j) \in T_k^2 : j \neq i,\ \mathrm{Cov}(e_i, e_j) = \omega$;

3. *all the remaining entries are identical*

   $\forall i \in T_k,\ \forall j \in T_\ell : \ell \neq k,\ \mathrm{Cov}(e_i, e_j) = \gamma$.

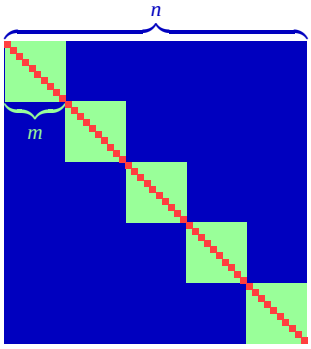


Figure 2: Structure of the covariance matrix.

Corollary 3 *The variance of the cross-validation estimator is a linear combination of three moments:*

$$\begin{aligned} \theta &= \frac{1}{n^2} \sum_{i,j} \mathrm{Cov}(e_i, e_j) \\ &= \frac{1}{n}\sigma^2 + \frac{m-1}{n}\omega + \frac{n-m}{n}\gamma \end{aligned} \tag{7}$$

Hence, the problem of estimating θ does not involve estimating $n(n+1)/2$ covariances, but it cannot be reduced to that of estimating a single variance parameter. Three components intervene, which may be interpreted as follows when $\hat{\mu}$ is the K-fold cross-validation estimate of EPE:

1. The variance σ^2 is the average (taken over training sets) variance of errors for "true" test examples when algorithm A is fed with training sets of size $m(K-1)$.

2. The within-block covariance ω would also apply to "true" test examples; it arises from the dependence of test errors stemming from the common training set.

3. The between-blocks covariance γ is due to the dependence of training sets (which share $n(K-2)/K$ examples) and the fact that test block T_k appears in all the training sets D_ℓ for $\ell \neq k$.

The forthcoming section makes use of this structure to show that there is no universal unbiased estimator of θ.

4. No Unbiased Estimator of $\mathrm{Var}[\hat{\mu}]$ Exists

Consider a generic estimator $\hat{\theta}$ that depends on the sequence of cross-validation errors $\mathbf{e} = (e_1, e_2, \ldots, e_n)'$. Let us assume that $\hat{\theta}$ is an analytic function of the errors, so that we can write its Taylor expansion

$$\hat{\theta} = \alpha_0 + \sum_i \alpha_1(i) e_i + \sum_{i,j} \alpha_2(i,j) e_i e_j + \sum_{i,j,k} \alpha_3(i,j,k) e_i e_j e_k + \ldots. \tag{8}$$

We first show that for unbiased variance estimates (i.e. $E[\hat{\theta}] = \mathrm{Var}[\hat{\mu}]$), all the α_i coefficients must vanish except for the second order coefficients $\alpha_{2,i,j}$.

Lemma 4 *There is no universal unbiased estimator of* $\mathrm{Var}[\hat{\mu}]$ *that involves the* e_i *in a non-quadratic way.*

Proof

Take the expected value of $\hat{\theta}$ *expressed as in (8), and equate it with* $\mathrm{Var}[\hat{\mu}]$ *(7):*

$$\begin{cases} E[\hat{\theta}] = \alpha_0 + \sum_i \alpha_1(i) E[e_i] + \sum_{i,j} \alpha_2(i,j) E[e_i e_j] + \sum_{i,j,k} \alpha_3(i,j,k) E[e_i e_j e_k] + \ldots \\ \theta = \frac{1}{n}\sigma^2 + \frac{m-1}{n}\omega + \frac{n-m}{n}\gamma. \end{cases}$$

For having $E[\hat{\theta}] = \theta$ *for all possible values of the moments of* $\mathbf{e}$*, one must have* $\alpha_0 = 0$ *because* θ *has no such constant term, not depending on any of the moments of* $\mathbf{e}$*. Similarly,* $\alpha_1(\cdot)$ *must be zero because* θ *has no term in* $E[e_i] = \mu$*. Finally, the third and higher order coefficients* $\alpha_\ell(\ldots)$*,* $\ell > 2$ *must also be zero because* θ *has only quantities depending on the second order moments* σ^2*,* ω *and* γ*.*

Q.E.D.

Since estimators that include moments other than the second moments in their expectation are biased, we now focus on the class of estimators which are quadratic forms of the errors, i.e.

$$\hat{\theta} = \mathbf{e}'\mathbf{W}\mathbf{e} = \sum_{i,j} W_{ij} e_i e_j. \tag{9}$$

Lemma 5 *The expectation of quadratic estimators* $\hat{\theta}$ *defined as in (9) is a linear combination of only three terms*

$$E[\hat{\theta}] = a(\sigma^2 + \mu^2) + b(\omega + \mu^2) + c(\gamma + \mu^2), \tag{10}$$

where (a, b, c) *are defined as follows:*

$$\begin{cases} a \stackrel{\Delta}{=} \sum_{i=1}^n W_{ii}, \\ b \stackrel{\Delta}{=} \sum_{k=1}^K \sum_{i \in T_k} \sum_{j \in T_k : j \neq i} W_{ij}, \\ c \stackrel{\Delta}{=} \sum_{k=1}^K \sum_{\ell \neq k} \sum_{i \in T_k} \sum_{j \in T_\ell} W_{ij}. \end{cases}$$

A "trivial" representer of estimators with this expected value is

$$\hat{\theta} = a s_1 + b s_2 + c s_3, \tag{11}$$

where (s_1, s_2, s_3) *are the only quadratic statistics of* $\mathbf{e}$ *that are invariants to the within blocks and between blocks permutations described in Lemma 1:*

$$\left\{\begin{array}{lcl} s_1 & \stackrel{\Delta}{=} & \frac{1}{n}\sum_{i=1}^{n} e_i^2, \\ s_2 & \stackrel{\Delta}{=} & \frac{1}{n(m-1)}\sum_{k=1}^{K}\sum_{i\in T_k}\sum_{j\in T_k: j\neq i} e_i e_j, \\ s_3 & \stackrel{\Delta}{=} & \frac{1}{n(n-m)}\sum_{k=1}^{K}\sum_{\ell\neq k}\sum_{i\in T_k}\sum_{j\in T_\ell} e_i e_j. \end{array}\right. \tag{12}$$

Proof
This result is obtained exploiting Corollary 2 and grouping the terms of $\hat{\theta}$ *in Equation (9) that have the same expected values.*

$$\begin{aligned} E[\hat{\theta}] &= \sum_{k=1}^{K}\sum_{i\in T_k}\left(W_{ii}E[e_i^2] + \sum_{j\in T_k: j\neq i} W_{ij}E[e_i e_j] + \sum_{\ell\neq k}\sum_{j\in T_\ell} W_{ij}E[e_i e_j]\right) \\ &= (\sigma^2+\mu^2)\sum_{i=1}^{n} W_{ii} + (\omega+\mu^2)\sum_{k=1}^{K}\sum_{i\in T_k}\sum_{j\in T_k: j\neq i} W_{ij} + \\ &\quad (\gamma+\mu^2)\sum_{k=1}^{K}\sum_{\ell\neq k}\sum_{i\in T_k}\sum_{j\in T_\ell} W_{ij} \\ &= a(\sigma^2+\mu^2) + b(\omega+\mu^2) + c(\gamma+\mu^2) \\ &= aE[s_1] + bE[s_2] + cE[s_3], \end{aligned}$$

which is recognized as the expectation of the estimator defined in Equation (11).

Q.E.D.

We now use Lemma 5 to prove that there is no *universally* unbiased estimator of $\mathrm{Var}[\hat{\mu}]$, i.e. there is no estimator $\hat{\theta}$ such that $E[\hat{\theta}] = \mathrm{Var}[\hat{\mu}]$ for all possible distributions of $\mathbf{e}$.

Theorem 6 *There exists no universally unbiased estimator of* $\mathrm{Var}[\hat{\mu}]$.
Proof
Because of Lemma 4 and 5, it is enough to prove the result for estimators that are quadratic forms expressed as in Equation (11). To obtain unbiasedness, the expected value of that estimator must be equated with $\mathrm{Var}[\hat{\mu}]$ *(7):*

$$a(\sigma^2+\mu^2) + b(\omega+\mu^2) + c(\gamma+\mu^2) = \frac{1}{n}\sigma^2 + \frac{m-1}{n}\omega + \frac{n-m}{n}\gamma. \tag{13}$$

For this equality to be satisfied for all distributions of cross-validation errors, it must be satisfied for all admissible values of μ, σ^2, ω, *and* γ. *This imposes the following unsatisfiable constraints on* (a,b,c)*:*

$$\left\{\begin{array}{lcl} a & = & \frac{1}{n}, \\ b & = & \frac{m-1}{n}, \\ c & = & \frac{n-m}{n}, \\ a+b+c & = & 0. \end{array}\right. \tag{14}$$

Q.E.D.

5. Eigenanalysis of the Covariance Matrix

One way to gain insight on the origin of the negative statement of Theorem 6 is via the eigenanalysis of Σ, the covariance of $\mathbf{e}$. This decomposition can be performed analytically thanks to the very particular block structure displayed in Figure 2.

Lemma 7 *Let $\mathbf{v}_k$ be the binary vector indicating the membership of each example to test block k. The eigensystem of Σ is as follows:*

- $\lambda_1 = \sigma^2 - \omega$ *with multiplicity* $n-K$ *and eigenspace defined by the orthogonal of basis* $\{\mathbf{v}_k\}_{k=1}^K$*;*
- $\lambda_2 = \sigma^2 + (m-1)\omega - m\gamma$ *with multiplicity* $K-1$ *and eigenspace defined in the orthogonal of* $\mathbf{1}$ *by the basis* $\{\mathbf{v}_k\}_{k=1}^K$*;*
- $\lambda_3 = \sigma^2 + (m-1)\omega + (n-m)\gamma$ *with eigenvector* $\mathbf{1}$*.*

Proof
From Corollary 2, the covariance matrix $\Sigma = E[\mathbf{e}\mathbf{e}'] - E[\mathbf{e}]E[\mathbf{e}]'$ can be decomposed as

$$\Sigma = (\sigma^2 - \omega)\Sigma_1 + m(\omega - \gamma)\Sigma_2 + m\gamma\Sigma_3,$$

where $\Sigma_1 = \mathbf{I}$, $\Sigma_2 = \frac{1}{m}(\mathbf{v}_1 \dots \mathbf{v}_K)(\mathbf{v}_1 \dots \mathbf{v}_K)'$ and $\Sigma_3 = \frac{1}{n}\mathbf{1}\mathbf{1}'$.

Σ_1, Σ_2 and Σ_3 share the same eigenvectors, with eigenvalues being equal either to zero or one:

- *the eigenvector $\mathbf{1}$ has eigenvalue 1 for Σ_1, Σ_2 and Σ_3;*
- *the eigenspace defined in the orthogonal of $\mathbf{1}$ by the basis $\{\mathbf{v}_k\}_{k=1}^K$ defines $K-1$ eigenvectors with eigenvalues 1 for Σ_1 and Σ_2 and 0 for Σ_3;*
- *all remaining eigenvectors have eigenvalues 1 for Σ_1 and 0 for Σ_2 and Σ_3.*

Q.E.D.

Lemma 7 states that the vector $\mathbf{e}$ can be decomposed into three uncorrelated parts: $n-K$ projections to the subspace orthogonal to $\{\mathbf{v}_k\}_{k=1}^K$, $K-1$ projections to the subspace spanned by $\{\mathbf{v}_k\}_{k=1}^K$ in the orthogonal of $\mathbf{1}$, and one projection on $\mathbf{1}$. A single vector example with n independent elements can be seen as n independent examples. Similarly, these projections of $\mathbf{e}$ can be equivalently represented by respectively $n-K$, $K-1$ and one uncorrelated one-dimensional examples, corresponding to the coordinates of $\mathbf{e}$ in these subspaces.

In particular, for the projection on $\mathbf{1}$, with only a single one-dimensional point, the sample variance is null, resulting in the absence of an unbiased variance estimator of λ_3. The projection of $\mathbf{e}$ on the eigenvector $\frac{1}{n}\mathbf{1}$ is precisely $\hat{\mu}$. Hence there is no unbiased estimate of $\mathrm{Var}[\hat{\mu}] = \frac{\lambda_3}{n}$ when we have only one realization of the vector $\mathbf{e}$. For the same reason, even with simple parametric assumptions on $\mathbf{e}$ (such as $\mathbf{e}$ Gaussian), the maximum likelihood estimate of θ is not defined. Only λ_1 and λ_2 can be estimated unbiasedly. Note that this problem cannot be addressed by performing multiple K-fold splits of the data set. Such a procedure would not provide independent realizations of $\mathbf{e}$.

6. Possible Values for ω and γ

Theorem 6 states that no estimator is unbiased, and in its demonstration, it is shown that the bias of any quadratic estimator is a linear combination of μ^2, σ^2, ω and γ. Regarding estimation, it is thus interesting to see what constraints restrict the possible range of these quantities.

Lemma 8 *For $\hat{\mu} = \mathrm{CV}$ and $\hat{\mu} = \Delta\mathrm{CV}$, the following inequalities hold:*

$$\left\{ \begin{array}{ccccc} 0 & \leq & \omega & \leq & \sigma^2 \\ -\frac{1}{n-m}(\sigma^2+(m-1)\omega) & \leq & \gamma & \leq & \frac{1}{m}(\sigma^2+(m-1)\omega) \end{array} \right.$$
$$\Rightarrow \left\{ \begin{array}{ccccc} 0 & \leq & \omega & \leq & \sigma^2 \\ -\frac{m}{n-m}\sigma^2 & \leq & \gamma & \leq & \sigma^2. \end{array} \right.$$

The shape of the admissible (ω,γ) region corresponding to the first set of (tighter) inequalities is displayed in Figure 3.

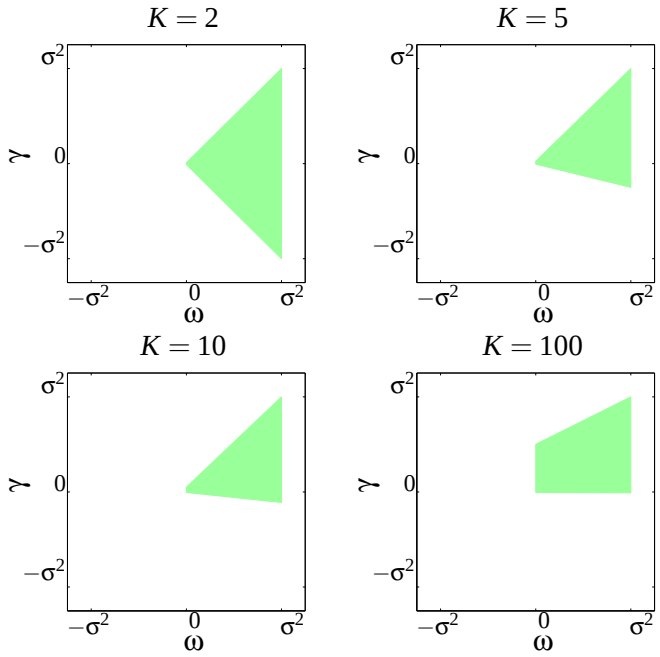


Figure 3: Possible values of (ω,γ) according to σ^2 for $n = 200$ and $K = \{2,5,10,100\}$.

Proof
The constraints on ω result from the Cauchy-Schwartz inequality which provides $\mathrm{Cov}(u,v)^2 \leq \mathrm{Var}[u]\mathrm{Var}[v]$, hence

$$-\sigma^2 \leq \omega \leq \sigma^2.$$

Moreover, the following reasoning shows that, for $\hat{\mu} = \mathrm{CV}$ and $\hat{\mu} = \Delta\mathrm{CV}$, ω is non-negative: ω is the covariance of (differences in) test errors for training sets of size $n-m$ and test sets of size $\ell = m$. The variance of the average test error is given by the mean of covariances $\frac{1}{\ell}(\sigma^2+(\ell-1)\omega)$. The variance σ^2 and covariance ω of test errors are not affected by ℓ, and the variance of the average

test error should be non-negative for any test set size ℓ. Hence ω is bound to be non-negative. When this type of reasoning cannot be used, as for $\hat{\mu} = \mathrm{JK}$, ω can only be proved to be greater than $-\sigma^2/(m-1)$.

The constraints on γ simply rephrase that the eigenvalues λ_2 and λ_3 of the covariance matrix Σ should be non-negative. The simpler (and looser) form is obtained by using $\omega \leq \sigma^2$.

Q.E.D.

The admissible (ω, γ) region obtained in Lemma 8 is very large. Furthermore, there is no constraint linking μ and σ^2, the mean and variance of e_i. Hence we cannot propose a variance estimate with universally small bias.

7. Experiments

We already mentioned that the bias of any quadratic estimator is a linear combination of μ^2, σ^2, ω and γ. The admissible values provided in the preceding section suggest that ω and γ cannot be proved to be negligible compared to σ^2. This section illustrates that in practice, the contribution to the variance of $\hat{\mu}$ due to ω and γ (see Equation (7)) can be of same order than the one due σ^2. It therefore suggests that the estimators of θ should indeed take into account the correlations of e_i.

Experiment 2 *True variance of K-fold cross-validation.*

We repeat the experimental setup of Experiment 1, except that now, we are in the more realistic situation where only one sample of size n is available. Since cross-validation is known to be sensitive to the instability of algorithms, in addition to this standard setup, we also consider another one with outliers:

The input $\mathbf{x_i} = (x_{i1}, \ldots, x_{id})'$ is still 30-dimensional, but it is now a mixture of two centered Gaussian variables: let t_i be a binary variable, with $P(t_i = 1) = p = 0.95$; when $t_i = 1$, $x_i \sim \mathcal{N}(0, \mathbf{I})$; when $t_i = 0$, $x_i \sim \mathcal{N}(0, 100\mathbf{I})$; $y_i = \sqrt{3/(d(p+100(1-p)))} \sum_{k=1}^{d} x_{ik} + \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(0, 1/(p+100(1-p)))$ when $t_i = 1$ and $\varepsilon_i \sim \mathcal{N}(0, 100/(p+100(1-p)))$ when $t_i = 0$.

We now look at the variance of K-fold cross-validation ($K = 10$), and decompose in the three orthogonal components σ^2, ω and γ. The results are shown in Figure 4.

When there are no outliers, the contribution of γ is very important for small sample sizes. For large sample sizes, the overall variance is considerably reduced and is mainly caused by σ^2. In these situations, the learning algorithm returns very similar answers for all training sets. When there are outliers, ω has little effect, but the contribution of γ is of same order as the one of σ^2, even when the ratio of examples to free parameters is large (here up to 20). Thus, in difficult situations, where $A(D)$ varies according to the realization of D, neglecting the effect of ω and γ can be expected to introduce a bias of the order of the true variance.

It is also interesting to see how these quantities are affected by the number of folds K. The decomposition of θ in σ^2, ω and γ (7) does not imply that K should be set either to n or to 2 (according to the sign of $\omega - \gamma$) in order to minimize the variance of $\hat{\mu}$. Modifying K affects σ^2, ω and γ through the size and overlaps of the training sets $D_1, \ldots, D_K$, as illustrated in Figure 5. For a fixed sample size, the variance of $\hat{\mu}$ and the contribution of σ^2, ω and γ effects varies smoothly with K.[6] The experiments with and without outliers illustrate that there is no general trend either in

6. Of course, the mean of $\hat{\mu}$ is also affected in the process.

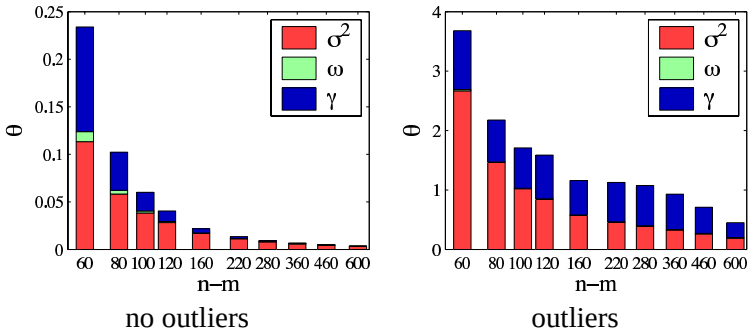


Figure 4: Bar plots of the contributions to total variance Var[CV] due to σ^2, ω and γ *vs.* the number of training examples $n-m$ for Experiment 2.

variance or decomposition of the variance in its σ^2, ω and γ components. The minimum variance can be reached for $K = n$ or for an intermediate value of K.

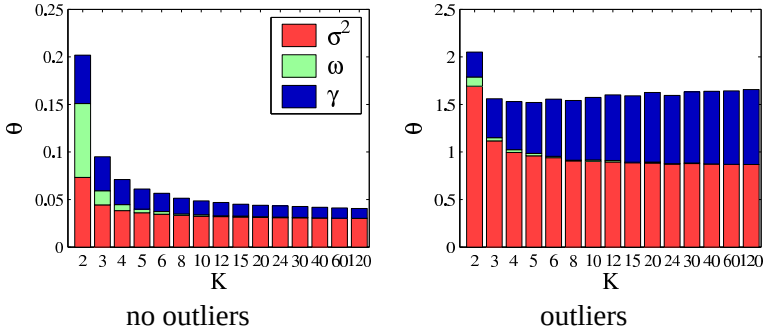


Figure 5: Bar plots of contributions of σ^2, ω and γ to θ *vs.* K for $n = 120$ for Experiment 2.

We also report an experiment illustrating that the previous observations also apply to classification on real data. The variance of K-fold cross-validation ($K = 10$), decomposed in the three orthogonal components σ^2, ω and γ is displayed in Figure 6.

Experiment 3 *Classification with trees on the Letter data set.*

The Letter data set comprises 20,000 examples described by 16 numeric features. The original setup considers 26 categories reprensenting the letters of the roman alphabet. Here, we used a simplified setup with 2 classes (A to M) vs. *(N to Z) in order to obtain sensible results for small sample sizes.*

Accurate estimates of σ^2, ω and γ require many independent training samples. This was achieved by considering the set of 20,000 examples to be the population, from which independent training samples were drawn by uniform sampling with replacement.

Here again, the variance of CV is mainly due to σ^2 and γ. According to the number of training examples, σ^2 is only responsible for 50 to 70 % of the total variance, so that a variance estimate based solely on σ^2 has a negative bias of the order of magnitude of the variance itself.

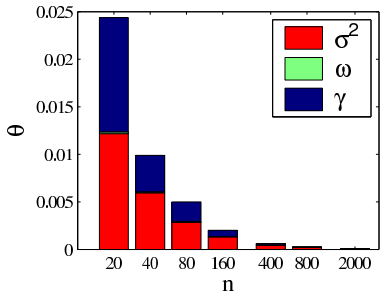


Figure 6: Bar plots of the contributions to total variance Var[CV] due to σ^2, ω and γ *vs.* the number of training examples n for Experiment 3.

8. Special Cases

This section addresses how our main result can be transposed to hold-out estimates of generalization error. We also detail how it applies to two specific instances of the general K-fold cross-validation scheme: two-fold and leave-one-out cross validation.

8.1 Hold-Out Estimate of EPE

When having *K independent training and test sets*, the structure of hold-out errors resemble the one of cross-validation errors, except that we know (from the independence of training and test sets) that $\gamma = 0$. This knowledge allows to build the unbiased variance estimate $\widehat{\theta_2}$ described in 2.2. This can be seen directly in the proof of Theorem 6: knowing that $\gamma = 0$ removes the third equation in the linear system (14). In practice, one is often restricted to $K = 1$ (ordinary hold-out test), which allows to estimate the variance due to the finite test set but not due to the particular choice of training set.

8.2 Two-Fold Cross-Validation

Two-fold cross-validation has been advocated to perform hypothesis testing (Dietterich, 1999; Alpaydin, 1999). It is a special case of K-fold cross-validation since the training blocks are mutually independent since they do not overlap. However, this independence does not modify the structure of $\mathbf{e}$ in the sense that γ is not null. The between-block correlation stems from the fact that the training block D_1 is the test block T_2 and vice-versa.

8.3 Leave-One-Out Cross-Validation

Leave-one-out cross-validation is a particular case of K-fold cross-validation, where $K = n$. The structure of the covariance matrix is simplified, without diagonal blocks: $\Sigma = (\sigma^2 - \gamma)\Sigma_1 + n\gamma\Sigma_3$. The estimation difficulties however remain: even in this particular case, there is no unbiased estimate of variance. From the definition of b (Lemma 5), we have $b = 0$, and with $m = 1$ the linear system (14) reads

$$\begin{cases} a & = & \frac{1}{n}, \\ c & = & \frac{n-1}{n}, \\ a+c & = & 0, \end{cases}$$

which still admits no solution.

9. Conclusions

It is known that K-fold cross-validation may suffer from high variability, which can be responsible for bad choices in model selection and erratic behavior in the estimated expected prediction error.

In this paper, we show that estimating the variance of K-fold cross-validation is difficult. Estimating a variance can be done from independent realizations or from dependent realizations whose correlation is known. K-fold cross-validation produces dependent test errors. Our analysis shows that although the correlations are structured in a very simple manner, their values cannot be estimated unbiasedly. Consequently, there is no unbiased estimator of the variance of K-fold cross-validation.

Our experimental section shows that in very simple cases, the bias incurred by ignoring the dependencies between test errors will be of the order of the variance itself. These experiments illustrate thus that the assessment of the significance of observed differences in cross-validation scores should be treated with much caution. The problem being unveiled, the next step of this study consists in building and comparing variance estimators dedicated to the very specific structure of the test error dependencies.

References

E. Alpaydin. Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, 11(8):1885–1892, 1999.

M. Anthony and S. B. Holden. Cross-validation for binary classification by real-valued functions: Theoretical analysis. In *Proceedings of the International Conference on Computational Learning Theory*, pages 218–229, 1998.

A. Blum, A. Kalai, and J. Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the International Conference on Computational Learning Theory*, pages 203–208, 1999.

L. Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383, 1996.

A. P. Dawid. Prequential analysis. In S. Kotz, C. B. Read, and D. L. Banks, editors, *Encyclopedia of Statistical Sciences, Update Volume 1*, pages 464–470. Wiley-Interscience, 1997.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1999.

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, 1993.

T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*, volume 43 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, 1990.

M. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, 1996.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1143, 1995.

C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52(3):239–281, 2003.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, B*, 36(1):111–147, 1974.

Journal of Machine Learning Research 5 (2004) 1107–1126 Submitted 12/03, Revised 4/04; Published 9/04

Selective Rademacher Penalization and Reduced Error Pruning of Decision Trees*

Matti Kääriäinen MATTI.KAARIAINEN@CS.HELSINKI.FI
Tuomo Malinen
Department of Computer Science, University of Helsinki
P.O. Box 68
FI-00014 Helsinki, Finland
Tapio Elomaa ELOMAA@CS.TUT.FI
Institute of Software Systems, Tampere University of Technology
P.O. Box 553
FI-33101 Tampere, Finland

Editor: Peter L. Bartlett

Abstract

Rademacher penalization is a modern technique for obtaining data-dependent bounds on the generalization error of classifiers. It appears to be limited to relatively simple hypothesis classes because of computational complexity issues. In this paper we, nevertheless, apply Rademacher penalization to the in practice important hypothesis class of unrestricted decision trees by considering the prunings of a given decision tree rather than the tree growing phase. This study constitutes the first application of Rademacher penalization to hypothesis classes that have practical significance. We present two variations of the approach, one in which the hypothesis class consists of all prunings of the initial tree and another in which only the prunings that are accurate on growing data are taken into account. Moreover, we generalize the error-bounding approach from binary classification to multi-class situations. Our empirical experiments indicate that the proposed new bounds outperform distribution-independent bounds for decision tree prunings and provide non-trivial error estimates on real-world data sets.

Keywords: generalization error analysis, data-dependent generalization error bounds, Rademacher complexity, decision trees, reduced error pruning

1. Introduction

Data-dependent bounds on generalization error of classifiers are bridging the gap that has existed between theoretical and empirical results since the introduction of computational learning theory. They allow to take situation specific information into account, whereas distribution-independent results need to hold in all imaginable situations. Using *Rademacher complexity* (Koltchinskii, 2001; Bartlett and Mendelson, 2002) to bound the generalization error of a training error minimizing classifier is a fairly new approach that has not yet been tested in practice extensively.

Rademacher penalization is in principle a general method applicable to any hypothesis class. However, in practice it does not seem amenable to complex hypothesis classes because the standard

*. This article is dedicated to the memory of the second author who unexpectedly passed away on June 6, 2004 at the age of twenty-seven.

method for computing Rademacher penalties relies on the existence of an empirical risk minimization algorithm for the hypothesis class in question. The first practical experiments with Rademacher penalization used real intervals as the hypothesis class (Lozano, 2000). Elomaa and Kääriäinen (2002) have applied Rademacher penalization to two-level decision trees, which can be learned efficiently in the agnostic PAC model (Auer et al., 1995).

General decision tree growing algorithms are necessarily heuristic because of the computational complexity of finding optimal decision trees (Grigni et al., 2000). Moreover, the hypothesis class consisting of unrestricted decision trees is so vast that traditional generalization error analysis techniques cannot provide non-trivial bounds for it. Nevertheless, top-down induction of decision trees by, e.g., C4.5 (Quinlan, 1993) produces results that are very competitive in prediction accuracy with better motivated approaches. We consider the usual two-phase process of decision tree learning; after growing a tree, it is pruned in order to reduce its dependency on the growing data and to better reflect characteristics of future data. Because of the practical success of decision tree learning, prunings of an induced decision tree can be considered an expressive class of hypotheses.

We apply Rademacher penalization to general decision trees by considering, not the tree growing phase, but rather the pruning phase. The idea is to view decision tree pruning as empirical risk minimization in the hypothesis class consisting of all prunings of an induced decision tree. First a heuristic tree growing procedure is applied to growing data to produce a decision tree. Then a pruning algorithm, for example the *reduced error pruning* (REP) algorithm of Quinlan (1987), is applied to the grown tree and a set of pruning data. As REP is known to be an efficient empirical risk minimization algorithm for the class of prunings of a decision tree (Elomaa and Kääriäinen, 2001), it can be used to compute the Rademacher penalty for this hypothesis class. Thus, by viewing decision tree pruning as empirical risk minimization in a data-dependent hypothesis class, we can bound the generalization error of prunings by Rademacher penalization. We also extend this generalization error analysis framework to the multi-class setting.

Standard Rademacher penalization still requires to take the whole hypothesis class into account. All possible prunings of the decision tree have to be evaluated. The prunings that evaluate best on randomly relabeled data—and, therefore, badly on the original data—essentially determine the error bound. However, in practice only prunings that have relatively small empirical error on the set of growing data are viable candidates for the final hypothesis. For this reason we restrict the pruning algorithm to operate on the much smaller class of hypotheses that consists of those prunings that make few mistakes on the set of growing data. To apply Rademacher penalization to this restricted class of hypotheses, we devise an empirical risk minimization algorithm for it. The new pruning algorithm, called k-REP, finds the most accurate pruning with respect to a set of pruning data among those prunings that make at most k mistakes on the set of growing data. The algorithm is based on dynamic programming and works in time cubic in the number of growing examples and linear in the number of pruning examples and the size of the decision tree to be pruned.

We evaluate the practical application potential of data-dependent error bounds empirically. Our experiments show that Rademacher penalization applied to prunings found by REP provides reasonable generalization error bounds on real-world data sets. The results for k-REP are even better. Although the bounds still overestimate the test set error, they are much tighter than distribution-independent bounds for prunings when the data sets are large.

This paper is organized as follows. In Section 2 we recapitulate the main idea of data-dependent generalization error analysis. We concentrate on Rademacher penalization, which we also extend to cover the multi-class case. Section 3 concerns pruning of decision trees, reduced error pruning

of decision trees being the main focus. The k-REP algorithm together with a correctness proof and time complexity analysis is presented in Section 4. Combining Rademacher complexity calculation and decision tree pruning is the topic of Section 5. Empirical evaluation of the proposed approach is presented in Section 6 and, finally, Section 7 presents the concluding remarks of this study.

2. Rademacher Penalties

Let $S = \{ (x_i, y_i) \mid i = 1, \ldots, n \}$ be a sample of n examples $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ each of which is drawn independently from some unknown probability distribution on $\mathcal{X} \times \mathcal{Y}$. In the PAC and statistical learning settings one usually assumes that the learning algorithm chooses its hypothesis $h \colon \mathcal{X} \to \mathcal{Y}$ from some fixed hypothesis class $\mathcal{H}$. Under this assumption generalization error analysis provides theoretical results bounding the generalization error of hypotheses $h \in \mathcal{H}$ which is allowed to depend on the sample, the learning algorithm, and the properties of the hypothesis class. We consider the multi-class setting, where $\mathcal{Y}$ may contain more than two labels.

Let P be the unknown probability distribution according to which the examples are drawn. The *generalization error* of a hypothesis h is the probability that a randomly drawn example (x, y) is misclassified:

$$\varepsilon_P(h) = P(h(x) \neq y).$$

The general goal of learning, of course, is to find a hypothesis with a small generalization error. However, since the generalization error depends on P, it cannot be computed directly based on the sample alone. We can try to approximate the generalization error of h by its *training error* on n examples:

$$\hat{\varepsilon}_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i),$$

where ℓ is the $0/1$ loss function

$$\ell(y, y') = \begin{cases} 1, & \text{if } y \neq y'; \\ 0, & \text{otherwise.} \end{cases}$$

Empirical Risk Minimization (ERM) (Vapnik, 1982) is a principle that suggest choosing the hypothesis $h \in \mathcal{H}$ with minimal training error. In relatively small and simple hypothesis classes finding a minimum training error hypothesis is computationally feasible. To guarantee that ERM yields hypotheses with small generalization error, one can try to bound $\sup_{h \in \mathcal{H}} |\varepsilon_P(h) - \hat{\varepsilon}_n(h)|$. Under the assumption that the examples are independent and identically distributed (i.i.d.), whenever $\mathcal{H}$ is not too complex, the difference of the training error of the hypothesis h on n examples and its true generalization error converges to 0 in probability as n tends to infinity.

The most common approach to deriving generalization error bounds is based on the VC dimension of the hypothesis class (Vapnik and Chervonenkis, 1971; Blumer et al., 1989). The problem with this approach is that it provides optimal results only in the worst case—when the underlying probability distribution is as bad as it can be. Thus, the generalization error bounds based on VC dimension tend to be overly pessimistic. Moreover, the VC dimension bounds are hard to extend to the multi-class setting. Data-dependent generalization error bounds, on the other hand, can be provably almost optimal for any given domain (Koltchinskii, 2001). In the following we review the foundations of a recent promising approach to bounding the generalization error.

A *Rademacher random variable* takes values $+1$ and -1 with probability 1/2 each. Let $r_1, r_2, \ldots, r_n$ be a sequence of Rademacher random variables independent of each other and the data $(x_1, y_1), \ldots, (x_n, y_n)$. The *Rademacher penalty* of the hypothesis class $\mathcal{H}$ is defined as

$$R_n(\mathcal{H}) = \sup_{h \in \mathcal{H}} \left| \frac{1}{n} \sum_{i=1}^{n} r_i \ell(h(x_i), y_i) \right|.$$

Rademacher penalty is, thus, a random variable depending both on the random choice of the learning sample $(x_1, y_1), \ldots, (x_n, y_n)$ and on the randomness injected through the random variables $r_1, \ldots, r_n$. The following symmetrization inequality (Van der Vaart and Wellner, 2000), which also covers the multi-class setting, connects Rademacher penalties to generalization error analysis.

Theorem 1 *The inequality*

$$\mathbf{E}\left[\sup_{h \in \mathcal{H}} |\varepsilon_P(h) - \hat{\varepsilon}_n(h)| \right] \leq 2\mathbf{E}[R_n(\mathcal{H})]$$

holds for any distribution P, number of examples n, and hypothesis class $\mathcal{H}$.

The random variables $\sup_{h \in \mathcal{H}} |\varepsilon_P(h) - \hat{\varepsilon}_n(h)|$ and $R_n(\mathcal{H})$ are sharply concentrated around their expectations (Koltchinskii, 2001). The concentration results are based on the following McDiarmid's (1989) bounded difference inequality.

Lemma 2 (McDiarmid's inequality) *Let $Z_1, \ldots, Z_n$ be independent random variables taking their values in a set A. Let $f : A^n \to \mathbb{R}$ be a function such that over all $z_1, \ldots, z_n, z_i' \in A$*

$$\sup |f(z_1, \ldots, z_i, \ldots, z_n) - f(z_1, \ldots, z_i', \ldots, z_n)| \leq c_i$$

for some constants $c_1, \ldots, c_n \in \mathbb{R}$. Then for all $\varepsilon > 0$

$$\mathbf{P}(f(Z_1, \ldots, Z_n) - \mathbf{E}[f(Z_1, \ldots, Z_n)] \geq \varepsilon) \text{ and}$$
$$\mathbf{P}(\mathbf{E}[f(Z_1, \ldots, Z_n)] - f(Z_1, \ldots, Z_n) \geq \varepsilon)$$

are upper bounded by

$$\exp\left(-2\varepsilon^2 \Big/ \sum_{i=1}^{n} c_i^2 \right).$$

Using McDiarmid's inequality one can bound the generalization error of hypotheses using their training error and Rademacher penalty as follows.

Lemma 3 *Let $h \in \mathcal{H}$ be arbitrary. Then with probability at least $1 - \delta$*

$$\varepsilon_P(h) \leq \hat{\varepsilon}_n(h) + 2R_n(\mathcal{H}) + 5\eta(\delta, n), \tag{1}$$

where $\eta(\delta, n) = \sqrt{\ln(2/\delta)/(2n)}$ is a hypothesis class independent error term that goes to zero as the number of examples increases.

Proof Observe that replacing a pair $((x_i,y_i),r_i)$ consisting of an example (x_i,y_i) and a Rademacher random variable r_i by any other pair $((x_i',y_i'),r_i')$ may change the value of $R_n(\mathcal{H})$ by at most $2/n$. Lemma 2 applied to the i.i.d. random variables $((x_1,y_1),r_1),\ldots,((x_n,y_n),r_n)$ and the function $R_n(\mathcal{H})$ yields

$$\mathbf{P}(R_n(\mathcal{H}) \leq \mathbf{E}[R_n(\mathcal{H})] - 2\eta(\delta,n)) \leq \frac{\delta}{2}. \tag{2}$$

Similarly, changing the value of any example (x_i,y_i) can change the value of $\sup_{h\in\mathcal{H}}|\varepsilon_P(h)-\hat{\varepsilon}_n(h)|$ by no more than $1/n$. Thus, applying Lemma 2 again to $(x_1,y_1),\ldots,(x_n,y_n)$ and $\sup_{h\in\mathcal{H}}|\varepsilon_P(h)-\hat{\varepsilon}_n(h)|$ gives

$$\mathbf{P}\left(\sup_{h\in\mathcal{H}}|\varepsilon_P(h)-\hat{\varepsilon}_n(h)| \geq \mathbf{E}\left[\sup_{h\in\mathcal{H}}|\varepsilon_P(h)-\hat{\varepsilon}_n(h)|\right]+\eta(\delta,n)\right) \leq \frac{\delta}{2}. \tag{3}$$

To bound the generalization error of a hypothesis $g\in\mathcal{H}$ observe that

$$\varepsilon_P(g) \leq \hat{\varepsilon}_n(g) + \sup_{h\in\mathcal{H}}|\varepsilon_P(h)-\hat{\varepsilon}_n(h)|.$$

Hence, by inequality (3), with probability at least $1-\delta/2$

$$\begin{aligned}\varepsilon_P(g) &\leq \hat{\varepsilon}_n(g)+\mathbf{E}\left[\sup_{h\in\mathcal{H}}|\varepsilon_P(h)-\hat{\varepsilon}_n(h)|\right]+\eta(\delta,n)\\ &\leq \hat{\varepsilon}_n(g)+2\mathbf{E}[R_n(\mathcal{H})]+\eta(\delta,n),\end{aligned}$$

where the second inequality follows from Theorem 1. Finally, applying inequality (2) yields that with probability at least $1-\delta$

$$\varepsilon_P(g) \leq \hat{\varepsilon}_n(g)+2R_n(\mathcal{H})+5\eta(\delta,n).$$

■

The usefulness of inequality (1) stems from the fact that its right-hand side depends only on the training sample and the Rademacher random variables, but not on P directly. Hence, all the data that is needed to evaluate the generalization error bound is available to the learning algorithm. Furthermore, Koltchinskii (2001) has shown that in the two-class situation the Rademacher penalty can be computed by an empirical risk minimization algorithm applied to relabeled training data. We now extend this method to the multi-class setting.

The expression for $R_n(\mathcal{H})$ is first written as the maximum of two suprema in order to remove the absolute value inside the original supremum:

$$R_n(\mathcal{H}) = \max\left(\sup_{h\in\mathcal{H}}\left\{\pm\frac{1}{n}\sum_{i=1}^{n}r_i\ell(h(x_i),y_i)\right\}\right).$$

The sum inside the supremum with positive sign is maximized by the hypothesis h_1 that tries to correctly classify those and only those training examples (x_i,y_i) for which $r_i=-1$. To formalize this, we associate each class $y\in\mathcal{Y}$ with a complement class label $\bar{y}$ that represents the set of all

classes but y. We denote the set of these complement classes by $\overline{\mathcal{Y}}$ and extend the domain of the loss function ℓ to cover pairs $(y,z) \in \mathcal{Y} \times \overline{\mathcal{Y}}$ by setting $\ell(y,z) = 1$ if $z = \bar{y}$ and 0 otherwise. Using this notation, h_1 is the hypothesis that minimizes the empirical error with respect to a newly labeled training set $\{(x_i, z_i)\}_{i=1}^n$, where

$$z_i = \begin{cases} y_i, & \text{if } r_i = -1; \\ \bar{y}_i, & \text{otherwise.} \end{cases}$$

The case for the supremum with negative sign is similar.

Altogether, the computation of the Rademacher penalty entails the following steps.

- Toss a fair coin n times to obtain a realization of the Rademacher random variable sequence $r_1, \ldots, r_n$.
- Change the label y_i to $\bar{y}_i$ if and only if $r_i = +1$ to obtain a new sequence of labels $z_1, \ldots, z_n$.
- Find functions $h_1, h_2 \in \mathcal{H}$ that minimize the empirical error with respect to the set of labels z_i and $\bar{z}_i$, respectively. Here, we follow the convention that $\bar{\bar{z}} = z$ for all $z \in \mathcal{Y} \cup \overline{\mathcal{Y}}$.
- Evaluate the Rademacher penalty given by the maximum of $|\{i : r_i = +1\}|/n - \hat{\varepsilon}(h_1)$ and $|\{i : r_i = -1\}|/n - \hat{\varepsilon}(h_2)$, where the empirical errors $\hat{\varepsilon}(h_1)$ and $\hat{\varepsilon}(h_2)$ are with respect to the labels z_i and $\bar{z}_i$, respectively.

In the two-class setting, the set $\bar{y}$ of all classes but y, $\mathcal{Y} \setminus \{y\}$, is a singleton. Thus, changing class y to $\bar{y}$ amounts to flipping the class label. It follows that a normal ERM algorithm can be used to find the hypotheses h_1 and h_2 and hence the Rademacher penalty can be computed efficiently provided that there exists an efficient ERM algorithm for the hypothesis class in question.

In the multi-class setting, however, a little more is required, since the sample on which the empirical risk minimization is performed may contain labels from $\overline{\mathcal{Y}}$ and the loss function differs from the standard $0/1$-loss. This, however, is not a problem with the variants of REP covered in this paper nor with T2, a decision tree learning algorithm used in our earlier study, since all the algorithms can be easily adapted to handle this more general setting. The case for REP is covered in the next sections and for T2 in the paper by Auer et al. (1995).

3. Growing and Pruning Decision Trees

A decision tree (Breiman et al., 1984) is a rooted tree in which the inner nodes are equipped with *branching functions* and the leaves are labeled with classes. A branching function routes examples reaching a node to its children, thus defining for each example a unique root-leaf path. The classification of an example is determined by the label of the leaf to which the example is routed.

A common approach in top-down induction of decision trees is to first grow a tree that fits the training data well and, then, prune it to reflect less the peculiarities of the training data; i.e., to generalize better. Here, pruning means replacing some inner nodes of the tree with leaves and removing the parts of the tree that become unreachable from the root. Many heuristic approaches (Quinlan, 1987; Mingers, 1989; Esposito et al., 1997) as well as more analytical ones (Mansour, 1997; Kearns and Mansour, 1998) to pruning have been proposed. A special class of pruning algorithms are the on-line ones (Helmbold and Schapire, 1997; Pereira and Singer, 1999). Even these algorithms work by the two-phase approach: An initial decision tree is fitted to the data and its prunings are then used as experts that collectively predict the class of observed instances.

Reduced error pruning was originally proposed by Quinlan (1987). It produces an optimal pruning of a given tree—the smallest tree among those with minimal error with respect to a given set of *pruning examples* (Esposito et al., 1997; Elomaa and Kääriäinen, 2001). The REP algorithm works in two phases: First the set of pruning examples S is classified using the given tree T to be pruned. Counters that keep track of the number of examples of each class passing through each node are updated simultaneously. In the second phase—a bottom-up pruning phase—those parts of the tree that can be removed without increasing the error of the remaining hypothesis are pruned away. The pruning decisions are based on the node statistics calculated in the top-down classification phase.

REP can be viewed as an ERM algorithm for the hypothesis class consisting of all prunings of a given decision tree. Thus, it can be used to efficiently compute Rademacher penalties and, hence, also generalization error bounds for the class of prunings of a decision tree. This leads us to the following strategy. First, we use a standard heuristic decision tree induction algorithm to grow a C4.5-type decision tree based on a set of growing examples. The tree serves as a representation of the data-dependent hypothesis class that consists of its prunings. As C4.5 usually performs quite well on real-world domains, it is reasonable to assume—even though it cannot be proved—that the class of prunings contains some good hypotheses.

Having grown a decision tree, we use a separate pruning data set to select one of the prunings of the grown tree as our final hypothesis. In this paper, we use REP as our pruning algorithm, but in principle any other pruning algorithm using the same basic pruning operation could be used instead. However, since REP is an empirical risk minimization algorithm, the derived error bounds will be the tightest when combined with the prunings produced by REP.

3.1 Reducing the Number of Prunings

As argued above, the set of prunings of a decision tree is likely to contain accurate hypotheses. Still, most of the prunings—the ones performing badly on the growing set—are likely to be very inaccurate on the pruning data. If the growing and the pruning data sets resemble each other to any extent, which is a necessary condition for the two-phase learning paradigm to make sense in the first place, the pruning algorithm will not select any of these hypotheses with very bad performance on the set of growing data. Keeping these inaccurate prunings as part of the hypothesis class only makes the hypothesis class more complex and, hence, increases the Rademacher penalty associated with it.

Following the line of thought above, it would seem reasonable to restrict the pruning algorithm to select the final pruning from among those hypotheses that are relatively accurate on the set of growing data. In Section 4 we present in detail the k-REP pruning algorithm, which does exactly this by solving the following problem: given a decision tree and sets of growing and pruning data, find the most accurate pruning (w.r.t. the pruning data) of the tree among those prunings that make at most k mistakes on the growing data. The restriction to prunings that are accurate on the growing data adds to the combinatorial complexity of the search problem, but we are still able to solve the problem in cubic time by using dynamic programming. k-REP is an efficient ERM algorithm for the restricted class of prunings. Thus, it can be used to evaluate generalization error bounds based on Rademacher penalties in the same way as REP can be used in connection with the class of all prunings (Kääriäinen and Elomaa, 2003). Since k-REP operates on a subclass of the class of all prunings, the Rademacher penalties are in this case smaller.

In order to use k-REP one has to devise some strategy of choosing a value for k, that is, to define exactly what it means for a hypothesis to be accurate on a set of growing data. If k is very large, k-REP boils down to standard REP since a loose bound on mistakes does not rule any of the prunings out. On the other hand, too small a k may shrink the hypothesis class too small or even empty if none of the prunings meets the strict accuracy requirement. A theoretically well-motivated solution would be to consider all values of k and employ standard model selection techniques to pick the one that gives the best error bounds. However, the model selection phase would loosen the bounds as the confidence parameter δ would have to be split among the different values of k. Hence, the best bound obtainable using model selection would unavoidably be larger than the best bound achievable if one could somehow pick a single fortunate choice for k.

In practice, the number of errors the original decision tree makes on the set of growing data is a good baseline to which the accuracy of the prunings can be related—we want the prunings considered by k-REP to be almost as accurate on the growing data as the original decision tree. Thus, we will select k to be some constant factor $c > 1$ times the number of errors the original tree makes on the growing data. This way of choosing the value of k is, of course, just an intuitively motivated heuristic, but so is the whole decision tree growing procedure that determines the original class of prunings in the first place. Our empirical experiments show this strategy works well on real world data sets.

A similar idea to the one behind k-REP is employed in the *shell decomposition bounds* of Langford and McAllester (2000), who show that the effective complexity of a hypothesis class can be measured by the complexity of the sub-class (or shell of hypothesis) that consists of only the almost most accurate hypotheses of the original class. The shells, however, are defined based on the same data that is used for selecting the final hypothesis, whereas in the case of k-REP the sub-class of accurate hypotheses is selected based on the growing data and the final hypothesis is chosen based on the pruning data. Also local Rademacher complexities (Bartlett et al., 2002, 2004; Lugosi and Wegkamp, 2004) and other local complexity measures (Koltchinskii and Panchenko, 2000; Massart, 2000; Mendelson and Philips, 2003) aim at taking into account only those parts of the model that are relevant for the given learning task. However, these methods have not been tested in practice as evaluating the local complexity measures involves some computational and other practical problems that have not been attacked yet.

3.2 Related Pruning Algorithms

REP produces the smallest of the most accurate prunings of a given decision tree, where accuracy is measured with respect to the pruning set. Other approaches for producing optimal prunings for different optimality criteria have also been proposed (Breiman et al., 1984; Bohanec and Bratko, 1994; Oliver and Hand, 1995; Almuallim, 1996). These criteria typically take both the size of the resulting pruning and its accuracy on growing data into account. As pruning tends to reduce growing set accuracy, one typically has to make a compromise between maintaining the initial growing set accuracy and finding a small pruning. For example, Bohanec and Bratko (1994) as well as Almuallim (1996) have studied how to efficiently find the smallest pruning that satisfies a given minimum accuracy requirement.

The strategy of using one data set for growing a decision tree and another for pruning it closely resembles the on-line pruning setting (Helmbold and Schapire, 1997; Pereira and Singer, 1999). In it the prunings of the initial decision tree are viewed as a pool of experts. Thus, pruning is

performed on-line, while giving predictions to new examples, rather than in a separate pruning phase. The main advantage of these on-line methods is that no statistical assumptions about the data generating process are needed and still the combined prediction and pruning strategy can be proved to be competitive with the best possible pruning of the initial tree. However, these approaches do not choose or maintain one pruning of the given decision tree, but rather a weighted combination of prunings, which may be impossible to interpret by human experts. Also, the loss bounds are meaningful only for very large data sets and there exists no empirical evaluation of the performance of the on-line pruning methods.

The pruning algorithms of Mansour (1997) and Kearns and Mansour (1998) are very similar to REP in spirit. The main difference with these algorithms and REP is the fact that they do not require the sample S on which pruning is based to be independent of the tree T; i.e., T may well have been grown based on S. Moreover, the pruning criterion in both methods is a kind of a *cost-complexity* condition (Breiman et al., 1984) that takes both the observed classification error and (sub)tree complexity into account. Both algorithms are *pessimistic*: They try to bound the true error of a (sub)tree by its training error. Since the training error is by nature optimistic, the pruning criterion has to compensate it by being pessimistic about the error approximation.

Both Mansour (1997) and Kearns and Mansour (1998) provide generalization error analyses for their algorithms. The bound presented in (Mansour, 1997) measures the complexity of the class of prunings by the size of the tree to be pruned. If this size or an upper bound for it is known in advance, the bound applies also when the pruning data is not independent of the tree to be pruned. Kearns and Mansour (1998) prove that the generalization error of the pruning produced by their algorithm is bounded by that of the best pruning of the given tree plus a complexity penalty. However, the penalty term can grow intolerably large and cannot be evaluated because of its dependence on the unknown optimal pruning and hidden constants.

One shortcoming of the two-phase decision tree induction approach is that there does not exist any well-founded approach for deciding how much data to use for the training and pruning phases. Only heuristic data set partitioning schemes are available. However, the simple rule of using, e.g., two thirds of the data for growing and the rest for pruning has been observed to work well in practice (Esposito et al., 1997). If the initial data set is very large, it may be computationally infeasible to use all the data for growing or pruning. In that case one can use heuristic sequential sampling methods for selecting the size of the growing set and determine the size of the pruning set, e.g., by using progressive Rademacher sampling (Elomaa and Kääriäinen, 2002). Because REP is an efficient linear-time algorithm, it is not hit hard by overestimated pruning sample size.

4. k-Optimal REP Prunings

Given a decision tree to be pruned and a set of pruning examples, REP finds the pruning that minimizes error on the pruning set; no consideration is given to the growing set error of the resulting hypothesis. In Section 3.1, we motivated the idea of imposing a restriction also on the growing set error of REP prunings. Clearly, in order to be able to prune at all, one has to give up some accuracy on the data that was used to grow the tree. This naturally leads to the idea of finding REP prunings with growing set error at most some threshold value k.

Let T be a (subtree of a) decision tree, $\hat{\varepsilon}_g(T)$ its growing set error, $\hat{\varepsilon}_p(T)$ its pruning set error, and $|T|$ its size. Let $\mathcal{P}(T)$ be the set of all the prunings of T.

Definition 4 *A k-optimal* REP *pruning of a decision tree T is a $T' \in \mathcal{P}(T)$ that has $\hat{\varepsilon}_g(T') \leq k$, if one exists, and for which*

- $\hat{\varepsilon}_p(T') = \min\{\hat{\varepsilon}_p(T'') \mid T'' \in \mathcal{P}(T),\ \hat{\varepsilon}_g(T'') \leq k\}$, *and*
- $|T'| = \min\{|T''| \mid T'' \in \mathcal{P}(T),\ \hat{\varepsilon}_p(T'') = \hat{\varepsilon}_p(T')\}$,

If there is no $T' \in \mathcal{P}(T)$ satisfying the criteria, k-optimal REP *pruning of T is undefined.*

For clarity, we consider only binary trees at first. Let T be a decision tree with root node N. Assume that, for each i, $0 \leq i \leq k$, we know i-optimal REP prunings of the children T_1 and T_2 of the root node N of T. Denote these by $T_1^0, \ldots, T_1^k$ and $T_2^0, \ldots, T_2^k$, respectively. Choosing any pair (T_1^u, T_2^v) of these prunings defines a pruning of T in the obvious way; let $\langle N, T_1^u, T_2^v \rangle$ denote this pruning.

In this paper we assume that leaf labels for decision tree prunings are determined by the growing data. Alternative leaf labeling strategies are discussed by Elomaa and Kääriäinen (2001) and a k-REP pruning algorithm resembling the one presented next could be derived for these labeling strategies as well. Let N_g denote the single-leaf pruning of T, i.e., a leaf labeled with the majority class of growing examples reaching T. The following result suggests a dynamic programming technique for finding k-optimal REP prunings, which is described subsequently.

Theorem 5 *If the k-optimal* REP *pruning of a decision tree T is defined, it is either the leaf N_g or of the form $\langle N, T_1^u, T_2^v \rangle$, where $u + v = k$ and T_1^u and T_2^v are u- and v-optimal* REP *prunings of the left and the right subtree of T, respectively.*

Proof Let T' be the k-optimal REP pruning of a decision tree T. If T' is N_g, then we have the claim. Otherwise, T' consists of a root node N and two subtrees T_1' and T_2', which respectively are prunings of the subtrees T_1 and T_2 of T. Now, $\hat{\varepsilon}_g(T') = \hat{\varepsilon}_g(T_1') + \hat{\varepsilon}_g(T_2') \leq k$, which means that there must exist u and v such that $u + v = k$, $\hat{\varepsilon}_g(T_1') \leq u$ and $\hat{\varepsilon}_g(T_2') \leq v$.

Let T_1^u be a u-optimal REP pruning of T_1 and assume that T_1' is not. By Definition 4 either $\hat{\varepsilon}_p(T_1') > \hat{\varepsilon}_p(T_1^u)$ or $|T_1'| > |T_1^u|$. Both cases contradict the k-optimality of pruning T', because the tree $\langle N, T_1^u, T_2' \rangle$ would be better than it. If $\hat{\varepsilon}_p(T_1') > \hat{\varepsilon}_p(T_1^u)$, then

$$\hat{\varepsilon}_p(T') = \hat{\varepsilon}_p(T_1') + \hat{\varepsilon}_p(T_2') > \hat{\varepsilon}_p(T_1^u) + \hat{\varepsilon}_p(T_2') = \hat{\varepsilon}_p(\langle N, T_1^u, T_2' \rangle).$$

If, on the other hand, $|T_1'| > |T_1^u|$, then

$$|T'| = |T_1'| + |T_2'| + 1 > |T_1^u| + |T_2'| + 1 = |\langle N, T_1^u, T_2' \rangle|.$$

Therefore, T_1' has to be a u-optimal REP pruning of T_1. Similar argumentation also proves the v-optimality of T_2'. ■

What Theorem 5 effectively says is that the k-optimal REP pruning of a tree T is either N_g or a combination of u- and v-optimal REP prunings of the children of its root node for some u and v summing up to k. Therefore, by going through each of the mentioned prunings, and minimizing over them first by pruning error, then by size, we can find k-optimal REP prunings of T. The k-optimal REP prunings are easy to find for trees consisting of single leafs. Combining this with a bottom

Algorithm 6 *Find k-optimal* REP *prunings.*

```
1   for each i ∈ {0,...,min(n,k)} do
2       ε̂_p(T^i) ← ∞;
3       |T^i| ← ∞
4   od;
5   if N is not a leaf then
6       for each i ∈ {0,...,min(n,k)} do
7           for each (u,v) such that u+v = i do
8               T' ← ⟨N, T_1^u, T_2^v⟩;
9               if ε̂_p(T') < ε̂_p(T^i) then T^i ← T' fi;
10              else if ε̂_p(T') = ε̂_p(T^i) and |T'| < |T^i| then T^i ← T' fi
11          od
12      od
13  fi;
14  for each i, i ∈ {ε̂_g(N_g),...,min(n,k)} do
15      if ε̂_p(N_g) ≤ ε̂_p(T^i) then T^i ← N_g fi
16  od;
```

up sweep of T yields a dynamic programming technique for the task at hand. The step of dynamic programming is given as Algorithm 6, which finds T^i for each i, $0 \leq i \leq \min(n,k)$, where n is the number of growing examples that reach the node. T^i is undefined for any i for which $|T^i| = \infty$ after running the algorithm.

The generalization of Theorem 5 (and Algorithm 6) to non-binary trees is straightforward. For a t-way split, one has to go through all the partitions of each i, $0 \leq i \leq \min(n,k)$, into t addends. This makes the time complexity exponential in the number of branches in the split, as the number of such partitions grows exponentially in t.

Let us consider the time complexity of Algorithm 6. Clearly, the loop on lines 14–16 works in time linear in $\min(n,k)$. In the loop on lines 6–12, one has to check i partitions for each i, $0 \leq i \leq \min(n,k)$. This makes the time complexity of processing a single node with a binary split $O(\min(n,k)^2)$, where n is the number of growing examples that reach the node.

Now consider a binary tree grown on n examples. First note that at most n growing examples reach the nodes of any particular level of the tree. Consider an arbitrary level with $w \leq n$ nodes, with $n_1, \ldots, n_w$ growing examples reaching them. By the above bound for a single node, the computation on the level takes $O(\sum_{i=1}^{w} \min(n_i,k)^2)$ steps. Now, it is clear that $\sum_{i=1}^{w} \min(n_i,k) \leq n$ holds, and this implies $\sum_{i=1}^{w} \min(n_i,k)^2 \leq n^2$, so $O(n^2)$ is an upper bound for the time complexity on any single level of the tree. A tree grown on n examples has at most n levels, which makes the worst case complexity $O(n^3)$.

The above result assumes that the pruning errors on lines 9, 10, and 15 can be evaluated in constant time. This can be achieved by equipping the nodes of the original tree with counters telling the class frequencies of pruning examples going through them. Initializing such counters can be done in time linear in the number of pruning examples and the size of the tree to be pruned. As the

algorithm does not need to access the pruning data after this preprocessing step, the time complexity with respect to the amount of pruning data is linear.

The $O(n^3)$ time complexity result can be strengthened if we make more assumptions on the decision tree to be pruned or the distribution of the growing examples to the tree. For example, if the depth of the tree can be assumed to be $O(\log n)$, the upper bound on the time complexity of k-REP is reduced to $O(n^2 \log n)$. As another special case, assume that the set of growing examples is halved in each node of a tree with n leaves. Then, the time complexity reduces to

$$c \sum_{i=0}^{\log n} 2^i \cdot \left(\min \left(\frac{n}{2^i}, k \right) \right)^2 = O(n^2),$$

where each addend corresponds to a single level of the tree.

5. Combining Rademacher Penalization and Decision Tree Pruning

When using REP or k-REP, the data sets used in growing the tree and pruning it are independent of each other. Therefore, any standard generalization error analysis technique can be applied to the resulting pruning as if the hypothesis class from which the pruning was selected was fixed in advance. A formal argument justifying this would be to carry out the generalization error analysis conditioned on the training data and then to argue that the bounds hold unconditionally by taking expectations over the selection of the training data set.

By the above argument, the theory of Rademacher penalization can be applied to the data-dependent class of prunings. Therefore, we can use the results presented in Section 2 to provide generalization error bounds for prunings found by REP, k-REP, or any other pruning algorithm. Moreover, since both REP and k-REP are efficient ERM algorithms (linear and cubic time, respectively) for the related classes of prunings, the generalization error bounds can be evaluated efficiently.

To summarize, we propose the following decision tree learning strategy that provides a generalization error bound for the hypothesis it produces:

1. Split the available data into a growing set and a pruning set.

2. Use, e.g., C4.5 (without pruning) on the growing set to induce a decision tree.

3. Find the smallest most accurate pruning of the tree built in the previous step using REP (or any other pruning algorithm) on the pruning set. This is the final hypothesis. Alternatively, choose a suitable k and use k-REP to find the most accurate pruning from the class of prunings making at most k errors on the set of growing data.

4. Evaluate the error bound as explained in Section 2 by running REP two more times. In case k-REP was used in step 3, use k-REP in place of REP here, too.

Even though the tree growing process is heuristic, the generalization error bounds for the prunings are provably true under the i.i.d. assumption. They are valid even if the tree growing heuristic fails, that is, when none of the prunings of the grown tree generalize well. In that case the bounds are, of course, unavoidably large. The situation is similar to, e.g., margin-based generalization error analysis (Cristianini and Shawe-Taylor, 2000), where the error bounds are good provided that the

training data generating distribution is such that a hypothesis with a good margin distribution can be found. In our case the error bounds are tight provided that C4.5 produces a decision tree that has good prunings and is still relatively small so that the Rademacher penalty for the class of its prunings does not blow up. A good choice of k may help in keeping the penalty term in control, a situation resembling choosing the marginal parameter in margin-based generalization error analysis. The existing empirical evidence overwhelmingly demonstrates that C4.5 usually fares quite well, and our experiments presented in Section 6 indicate that a good choice of k really results in a notable decrease in the complexity term on real world data sets.

The value of k should ideally be so large that the hypothesis class associated with it includes the most accurate pruning w.r.t. pruning data and, at the same time, as small as possible to limit the complexity of the remaining hypothesis class to a minimum. This trade-off is hard to solve in general, since the decision on which k to choose has to be done prior to seeing the set of pruning data. In the following we will choose k to be some $c > 1$ times the number of errors the original decision tree makes on the set of growing data. This way we take into account the fact that the original tree most likely overfits the growing data set and thus has a smaller error than can be expected from prunings with good generalization. The empirical experiments indicate that $c = 1.1$ is a reasonable choice for all data sets we experimented with.

Generalization error bounds can be roughly divided into two categories: Those based on a training set only and those requiring a separate test set (Langford, 2002). Our generalization error bounds for prunings may be seen to lie somewhere between these two extremes, the bound for k-REP being the one closer to test set bounds. We use only part of the data in the tree growing phase that determines our hypothesis class. The rest—the set of pruning data—is used only for selecting a pruning and evaluating the generalization error bound. Thus, some of the information contained in the pruning set may be lost as it cannot be used in the tree induction phase. However, the pruning set is still used for the non-trivial task of selecting a good pruning, so that some of the information contained in it can be exploited in the final hypothesis. The pruning set is thus used as a test set for the outcome of the tree growing phase and also as a proper learning set in the pruning phase.

6. Empirical Evaluation

Before reporting and discussing the results obtained in our tests, we describe the distribution-independent bound used as comparison point to Rademacher penalization and briefly outline other aspects of the test setting.

6.1 Test Setting for Performance Comparison

The obvious performance reference for Rademacher penalization over decision tree prunings is to compare it to existing generalization error bounds. The bound of Kearns and Mansour (1998) is impossible to evaluate in practice because it requires knowing the depth and size of the pruning with the best generalization error. The bound presented by Mansour (1997) only requires knowing the maximum size of prunings in advance and would, thus, be applicable in our setting. However, Mansour's bound is clearly inferior to the simpler Occam's Razor type of bound to be introduced next and will, hence, be excluded from the empirical comparison. Bounds developed in the on-line pruning setting (Helmbold and Schapire, 1997) are incomparable with the one presented in this paper because of the different learning model. Thus, they will not be considered here.

The simplest—and as it turned out in our experiments, the tightest—existing generalization error bound which the Rademacher bound can be compared to is to our knowledge an Occam's Razor bound (Blumer et al., 1987; Langford, 2003) that is obtained by assigning equal-length codes to all prunings of the original decision tree. Equivalently, we assign equal prior probability to all prunings of the original tree. Since the leaf labels of the prunings are determined by the growing data, all that needs to be encoded is the set of those inner nodes that are to be replaced by leaves. A simple way to do so is to assign a bit for each of the $(d-1)/2$ inner nodes of a d node tree telling whether the node is pruned or not.

The simplistic code outlined above contains some redundancy as, e.g., the pruning consisting of a single leaf is represented by $2^{(d-1)/2-1}$ different codewords. However, it is easy to see that a binary tree with d nodes can have at least $2^{d/4}$ prunings; consider, e.g., the prunings obtainable from a balanced tree by pruning a subset of inner nodes next to the leaves. Thus, no less than $d/4$ bits will suffice if nothing but the size of the tree to be pruned is taken into account. To find out the optimal uniform code length given the whole tree to be pruned as a parameter, one would essentially have to count the number of prunings of the tree. We are not aware of an efficient algorithm for this task. On the other hand, using a non-uniform code length would introduce a bias to the bound that is not present in our proposed bounds. Thus, in our experiments we will use the code length approximation $d/4$, giving worst-case optimistic error bounds. Plugging this into the Chernoff Occam's Razor bound (Langford, 2003) we get that with probability at least $1-\delta$,

$$\varepsilon_P(h) < \hat{\varepsilon}_n(h) + \sqrt{\frac{\ln 2 \cdot d/4 + \ln(1/\delta)}{2n}},$$

where d is the number of the nodes of the tree and n is the size of the pruning set. This bound could be further improved by using the exact Occam's razor bound (Langford, 2003) instead, but we have not tried how significant the improvement would be. Note that this bound is data independent in the sense that the pruning data is taken into account only through the pruning error $\hat{\varepsilon}_n(h)$.

The error bounds based on Rademacher penalization depend on the data distribution so that their true performance can be evaluated only empirically. In our experiments we grow binary decision trees using a C4.5-type decision tree algorithm distributed in the Weka package (Witten and Frank, 1999). As a benchmark we use 15 data sets from the UCI Machine Learning Repository (Blake and Merz, 1998). In each experiment we allocate 10 percent of the data for testing and split the rest to growing and pruning sets. As the split ratio we chose 2:1 as suggested by Esposito et al. (1997). For the generated data set LED, we use 300,000 instances with 10 percent attribute noise. For k-REP we choose $c = 1.1$, i.e., k is 1.1 times the training error of the unpruned tree.

6.2 Empirical Observations

Table 1 and Figure 1 summarize the results over 10 random splits of the data sets. In Table 1 we present the decision tree sizes before and after pruning with k-REP and REP. Observe that the unpruned decision trees are very large, which means that the class of prunings may potentially be very complex. The results indicate that REP manages to decrease the tree sizes considerably. The sizes of k-REP prunings fall in many cases roughly halfway between the unpruned tree size and the size of the REP pruned tree.

Figure 1 presents the test set accuracies and error bounds based on Rademacher penalization and Occam's Razor. In all bounds, we set $\delta = 0.01$. Even though both bounds based on Rademacher

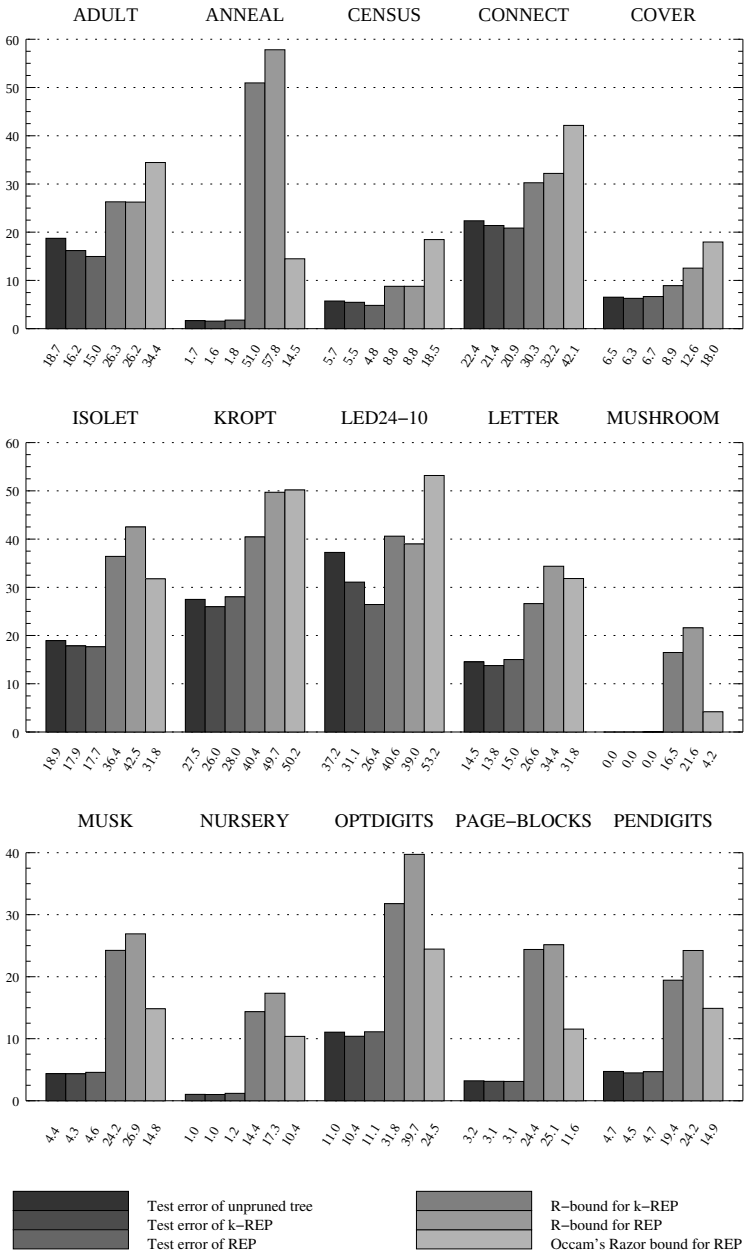


Figure 1: Averages of error bounds over 10 random splits of the data sets.

DATA SET	UNPRUNED	*k*-REP	REP
ADULT	7,507.6	3,898.6	1,600.6
ANNEAL	32.0	24.8	20.8
CENSUS	20,513.4	12,378.6	4,819.4
CONNECT	13,953.8	8,583.8	4,289.0
COVER	31,483.6	25,374.0	18,396.4
ISOLET	664.8	517.6	272.0
KROPT	7,317.4	5,328.8	3,572.4
LED24-10	90,564.8	43,689.4	9,041.6
LETTER	2,543.8	1,907.0	1,292.4
MUSHROOM	22.8	22.8	22.0
MUSK	224.8	186.0	120.0
NURSERY	392.0	349.4	306.8
OPTDIGITS	410.4	319.8	222.2
PAGE-BLOCKS	123.2	85.6	42.6
PEN-DIGITS	411.0	324.0	245.8

Table 1: Average sizes of trees over 10 random splits of the data sets.

penalization clearly overshoot the test set accuracies, they still provide reasonable estimates in many cases. Note that in the multi-class settings even error bounds above 50 percent are non-trivial.

Both bounding methods, the one based on Rademacher penalties and the one based on Occam's Razor, outperform the other on a number of data sets; there seems to be no clear overall winner. Notably, in many cases the difference between the better and worse method is quite large. On large data sets, the Rademacher bounds are consistently better; the converse holds for the small sets. The small amount of data blows up the hypothesis class independent term $\eta(\delta, n)$ to the extent that it starts to dominate the actual Rademacher penalty. The Occam's Razor bound is clearly better when the unpruned tree is small, since this situation keeps the penalty term related to it under control.

Rademacher bounds for *k*-REP turn out to be better than the REP bound in most cases. The only notable exception is the LED domain, where the pruning error of the best pruning is significantly lower than that of the best restricted pruning, while the Rademacher penalties for both classes are almost the same. In CENSUS INCOME the decrease of pruning error and growth of the Rademacher penalty cancel each other out so that the bounds for REP and *k*-REP are nearly equal.

We also conducted a set of experiments in order to see how the bound behaves as a function of c. The results indicate that decreasing c typically yields tighter bounds, but at the same time the actual quality of the prunings obtained deteriorates as c gets closer to 1. In the limiting case $c = 1$ there is no room left for pruning, so this extreme case effectively coincides with using the pruning set as a set of test data. Increasing c relaxes the restrictions on the pruning decisions and enables *k*-REP to find prunings with better empirical performance. The trade-off here is a special case of the fact that test error bounds are typically the tightest in practice even though using all the data in learning might yield a hypothesis with better generalization error. Our choice of $c = 1.1$ seems to be a good compromise between the tightness of the bound and the actual generalization performance of the obtained pruning.

The relative test performance of *k*-REP and REP is varied and neither method seems to be a clear winner. As *k*-REP produces larger prunings and is computationally more demanding than REP, there

seems to be little motivation for using k-REP independently as a pruning method if error guarantees are not called for.

Our intention has been to carry out a feasibility study of the new technique of Rademacher penalization, rather than to aim at generalization error bounds directly applicable in the real world. However, the bounds that were obtained on larger data sets are sometimes tighter than one could have expected in advance. In the best cases the theoretical bounds already approach usability as performance guarantees of practical algorithms. Even though even the best of the proposed bounds always overestimates the test error, it is never totally unrealistic. Thus, we have demonstrated that Rademacher penalization represents a step toward the use of well-founded training set bounds in practical applications. Though, at the same time it is, unfortunately, not possible to draw too far-reaching positive conclusions from this study, because in the worst cases Rademacher penalization fails to deliver usable bounds and does not fare as well as the Occam's Razor bound on smaller data sets.

7. Conclusion

Modern generalization error bounding techniques that take the observed data distribution into account give far more realistic sample complexities and generalization error approximations than the distribution-independent methods. We have shown how one of these techniques, namely Rademacher penalization, can be applied to bound the generalization error of decision tree prunings, also in the multi-class setting. According to our empirical experiments the proposed theoretical bounds are often tighter than distribution-independent generalization error bounds for decision tree prunings. However, the new bounds still appear unable to faithfully describe the performance attained in practice.

As future work, we intend to carry out more thorough empirical experiments on the proposed methods. Also, we will look for better motivated ways of tuning the value of c and of determining the proportion of learning data allocated for pruning purposes. It would also be interesting to extend the two-phase generalization error analysis approach introduced here to other hypothesis classes, too.

Acknowledgments

We thank the anonymous referees for suggestions that improved the final version of this paper; in particular, for pointing out the possibility of using the Occam's Razor bound as the comparison point in our empirical evaluation.

References

Hussein Almuallim. An efficient algorithm for optimal pruning of decision trees. *Artificial Intelligence*, 83(2):347–362, 1996.

Peter Auer, Robert C. Holte, and Wolfgang Maass. Theory and application of agnostic PAC-learning with small decision trees. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 21–29, San Francisco, CA, 1995. Morgan Kaufmann.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Localized Rademacher complexities. In Jyrki Kivinen and Robert H. Sloan, editors, *Computational Learning Theory, Proceedings of the Fifteenth Annual Conference*, volume 2375 of *Lecture Notes in Artificial Intelligence*, pages 44–58, Berlin Heidelberg New York, 2002. Springer.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local Rademacher complexities. *The Annals of Statistics*, 2004. To appear.

Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

Cathrine L. Blake and Christopher J. Merz. *UCI Repository of Machine Learning Databases*. University of California, Department of Information and Computer Science, Irvine, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

Marco Bohanec and Ivan Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Pacific Grove, 1984.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

Tapio Elomaa and Matti Kääriäinen. An analysis of reduced error pruning. *Journal of Artificial Intelligence Research*, 15:163–187, 2001.

Tapio Elomaa and Matti Kääriäinen. Progressive Rademacher sampling. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 140–145, Cambridge, MA, 2002. MIT Press.

Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (5):476–491, 1997.

Michelangelo Grigni, Vincent Mirelli, and Christos H. Papadimitriou. On the difficulty of designing good classifiers. *SIAM Journal on Computing*, 30(1):318–323, 2000.

David P. Helmbold and Robert E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, 1997.

Matti Kääriäinen and Tapio Elomaa. Rademacher penalization over decision tree prunings. In Nada Lavrač, Dragan Gamberger, Hendrik Blockeel, and Ljupčo Todorovski, editors, *Machine Learning: ECML 2003, Proceedings of the Fourteenth European Conference*, volume 2837 of *Lecture Notes in Artificial Intelligence*, pages 193–204, Berlin Heidelberg New York, 2003. Springer.

Michael Kearns and Yishay Mansour. A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In Jude Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 269–277, San Francisco, CA, 1998. Morgan Kaufmann.

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.

Vladimir Koltchinskii and Dmitry Panchenko. Rademacher processes and bounding the risk of function learning. In Evarist Giné, David M. Mason, and Jon A. Wellner, editors, *High Dimensional Probability II*, pages 443–459. Birkhäuser, Boston, 2000.

John Langford. Combining training set and test set bounds. In Claude Sammut and Achim G. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 331–338, San Francisco, CA, 2002. Morgan Kaufmann.

John Langford. Practical prediction theory for classification, 2003. A tutorial presented at ICML 2003. Available at `http://hunch.net/~jl/projects/prediction_bounds/tutorial/tutorial.pdf`.

John Langford and David McAllester. Computable shell decomposition bounds. In Nicolò Cesa-Bianchi and Sally A. Goldman, editors, *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 25–34, San Francisco, CA, 2000. Morgan Kaufmann.

Fernando Lozano. Model selection using Rademacher penalization. In *Proceedings of the Second ICSC Symposium on Neural Networks*, Berlin, 2000. NAISO Academic Press.

Gábor Lugosi and Marten Wegkamp. Complexity regularization via localized random penalties. *The Annals of Statistics*, 32(4):1679–1697, 2004.

Yishay Mansour. Pessimistic decision tree pruning based on tree size. In Douglas H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 195–201, San Francisco, CA, 1997. Morgan Kaufmann.

Pascal Massart. Some applications of concentration inequalities to statistics. *Annales de la Faculté des Sciences de Toulouse*, IX:245–303, 2000.

Colin McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, volume 141 of *London Mathematical Society Lecture Note Series*, pages 148–188. Cambridge University Press, 1989.

Shahar Mendelson and Petra Philips. Random subclass bounds. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines, Proceedings of the Sixteenth Annual Conference on Learning Theory and Seventh Kernel Workshop, COLT/Kernel 2003*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 329–343, Berlin Heidelberg New York, 2003. Springer.

John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, 1989.

Jonathan J. Oliver and David J. Hand. On pruning and averaging decision trees. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 430–437, San Francisco, CA, 1995. Morgan Kaufmann.

Francesco C. Pereira and Yoram Singer. An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning*, 36(3):183–199, 1999.

J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3): 221–248, 1987.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

Aad W. Van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes*. Springer, New York, 2000. Corrected second printing.

Vladimir N. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer, New York, 1982.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280, 1971.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA, 1999.

Journal of Machine Learning Research 5 (2004) 1127–1141 Submitted 12/03; Revised 7/04; Published 9/04

Knowledge-Based Kernel Approximation

Olvi L. Mangasarian OLVI@CS.WISC.EDU
Jude W. Shavlik SHAVLIK@CS.WISC.EDU
Edward W. Wild WILDT@CS.WISC.EDU
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, WI 53706, USA

Editor: John Shawe-Taylor

Abstract

Prior knowledge, in the form of linear inequalities that need to be satisfied over multiple polyhedral sets, is incorporated into a function approximation generated by a linear combination of linear or nonlinear kernels. In addition, the approximation needs to satisfy conventional conditions such as having given exact or inexact function values at certain points. Determining such an approximation leads to a linear programming formulation. By using nonlinear kernels and mapping the prior polyhedral knowledge in the input space to one defined by the kernels, the prior knowledge translates into nonlinear inequalities in the original input space. Through a number of computational examples, including a real world breast cancer prognosis dataset, it is shown that prior knowledge can significantly improve function approximation.

Keywords: function approximation, regression, prior knowledge, support vector machines, linear programming

1. Introduction

Support vector machines (SVMs) play a major role in classification problems (Vapnik, 2000, Cherkassky and Mulier, 1998, Mangasarian, 2000). More recently, prior knowledge has been incorporated into SVM classifiers, both to improve the classification task and to handle problems where conventional data may be few or not available (Schölkopf et al., 1998, Fung et al., 2003b,a). Although SVMs have also been extensively used for regression (Drucker et al., 1997, Smola and Schölkopf, 1998, Evgeniou et al., 2000, Mangasarian and Musicant, 2002), prior knowledge on properties of the function to be approximated has not been incorporated into the SVM function approximation as has been done for an SVM classifier (Fung et al., 2003b,a). In this work, we introduce prior knowledge in the form of linear inequalities to be satisfied by the function on polyhedral regions of the input space for linear kernels, and on similar regions of the feature space for nonlinear kernels. These inequalities, unlike point-wise inequalities or general convex constraints that have already been treated in approximation theory (Mangasarian and Schumaker, 1969, 1971, Micchelli and Utreras, 1988, Deutsch, 2001), are inequalities that need to be satisfied over specific polyhedral sets. Such "prior knowledge" does not seem to have been treated in the extensive approximation theory literature.

We outline the contents of the paper now. In Section 2 we define the prior knowledge formulation for a linear kernel approximation in the input space of the problem which leads to a linear

programming formulation in that space. In Section 3 we approximate the function by a linear combination of nonlinear kernel functions and explicitly map the polyhedral prior knowledge in the input space to one defined by the kernel functions. This leads to a linear programming formulation in that space. In Section 4 we demonstrate the utility of our results on a number of synthetic approximation problems as well as a real world breast cancer prognosis dataset where we show that prior knowledge can improve the approximation. Section 5 concludes the paper with a brief summary and some possible extensions and applications of the present work.

We describe our notation now. All vectors will be column vectors unless transposed to a row vector by a prime $'$. The scalar (inner) product of two vectors x and y in the n-dimensional real space R^n will be denoted by $x'y$. For $x \in R^n$, $\|x\|_1$ denotes the 1-norm: $\sum_{i=1}^{n} |x_i|$. The notation $A \in R^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix, A' will denote the transpose of A, A_i will denote the i-th row of A and $A_{\cdot j}$ the j-th column of A. A vector of ones in a real space of arbitrary dimension will be denoted by e. Thus for $e \in R^m$ and $y \in R^m$ the notation $e'y$ will denote the sum of the components of y. A vector of zeros in a real space of arbitrary dimension will be denoted by 0. For $A \in R^{m\times n}$ and $B \in R^{n\times k}$, a *kernel* $K(A,B)$ maps $R^{m\times n} \times R^{n\times k}$ into $R^{m\times k}$. In particular, if x and y are column vectors in R^n then, $K(x',y)$ is a real number, $K(x',A')$ is a row vector in R^m and $K(A,A')$ is an $m \times m$ matrix. We shall make no assumptions on our kernels other than symmetry, that is $K(x',y)' = K(y',x)$, and in particular we shall not assume or make use of Mercer's positive semidefiniteness condition (Vapnik, 2000, Schölkopf and Smola, 2002). The base of the natural logarithm will be denoted by ε. A frequently used kernel in nonlinear classification is the Gaussian kernel (Vapnik, 2000, Cherkassky and Mulier, 1998, Mangasarian, 2000) whose ijth element, $i = 1\ldots,m,\ j = 1\ldots,k$, is given by: $(K(A,B))_{ij} = \varepsilon^{-\mu\|A_i' - B_{\cdot j}\|^2}$, where $A \in R^{m\times n}$, $B \in R^{n\times k}$ and μ is a positive constant. Approximate equality is denoted by $\approx$, while the abbreviation "s.t." stands for "subject to". The symbol $\wedge$ denotes the logical "and" while $\vee$ denotes the logical "or".

2. Prior Knowledge for a Linear Kernel Approximation

We begin with a linear kernel model and show how to introduce prior knowledge into such an approximation. We consider an unknown function f from R^n to R for which approximate or exact function values are given on a dataset of m points in R^n denoted by the matrix $A \in R^{m\times n}$. Thus, corresponding to each point A_i we are given an exact or inexact value of f, denoted by a real number y_i, $i = 1,\ldots,m$. We wish to approximate f by some linear or nonlinear function of the matrix A with unknown linear parameters. We first consider the simple linear approximation

$$f(x) \approx w'x + b, \tag{1}$$

for some unknown weight vector $w \in R^n$ and constant $b \in R$ which is determined by minimizing some error criterion that leads to

$$Aw + be - y \approx 0. \tag{2}$$

If we consider w to be a linear combination of the rows of A, i.e. $w = A'\alpha$, $\alpha \in R^m$, which is similar to the dual representation in a linear support vector machine for the weight w (Mangasarian, 2000, Schölkopf and Smola, 2002), we then have

$$AA'\alpha + be - y \approx 0. \tag{3}$$

This immediately suggests the much more general idea of replacing the linear kernel AA' by some arbitrary nonlinear kernel $K(A,A') : R^{m\times n} \times R^{n\times m} \longrightarrow R^{m\times m}$ that leads to the following approximation, which is nonlinear in A but linear in α:

$$K(A,A')\alpha + be - y \approx 0. \tag{4}$$

We will measure the error in (4) componentwise by a vector $s \in R^m$ defined by

$$-s \leq K(A,A')\alpha + be - y \leq s. \tag{5}$$

We now drive this error down by minimizing the 1-norm of the error s together with the 1-norm of α for complexity reduction or stabilization. This leads to the following constrained optimization problem with positive parameter C that determines the relative weight of exact data fitting to complexity reduction:

$$\begin{array}{rcl} & \min\limits_{(\alpha,b,s)} & \|\alpha\|_1 + C\|s\|_1 \\ \text{s.t.} \quad -s \quad \leq & K(A,A')\alpha + be - y & \leq \quad s, \end{array} \tag{6}$$

which can be represented as the following linear program:

$$\begin{array}{rcccl} & & \min\limits_{(\alpha,b,s,a)} & e'a + Ce's & \\ \text{s.t.} & -s \leq & K(A,A')\alpha + be - y & \leq & s, \\ & -a \leq & \alpha & \leq & a. \end{array} \tag{7}$$

We note that the 1-norm formulation employed here leads to a linear programming formulation without regard to whether the kernel $K(A,A')$ is positive semidefinite or not. This would not be the case if we used a kernel-induced norm on α that would lead to a quadratic program. This quadratic program would be more difficult to solve than our linear program especially when it is nonconvex, which would be an NP-hard problem, as is the case when the kernel employed is not positive semidefinite.

We now introduce prior knowledge for a linear kernel as follows. Suppose that it is known that the function f represented by (1) satisfies the following condition. For all points $x \in R^n$, not necessarily in the training set but lying in the nonempty polyhedral set determined by the linear inequalities

$$Bx \leq d, \tag{8}$$

for some $B \in R^{k\times n}$, the function f, and hence its linear approximation $w'x + b$, must dominate a given linear function $h'x + \beta$, for some user-provided $(h,\beta) \in R^{n+1}$. That is, for a *fixed* (w,b) we have the implication

$$Bx \leq d \implies w'x + b \geq h'x + \beta, \tag{9}$$

or equivalently in terms of α, where $w = A'\alpha$:

$$Bx \leq d \implies \alpha'Ax + b \geq h'x + \beta. \tag{10}$$

Thus, the implication (10) needs to be added to the constraints of the linear program (7). To do that we make use of the following equivalence relationship that converts the implication (10) to a set of linear constraints that can be appended to the linear program (7). A similar technique was used in (Fung et al., 2003b, Proposition 2.1) to incorporate prior knowledge into linear classifiers.

Proposition 2.1 Prior Knowledge Equivalence. *Let the set* $\{x \mid Bx \leq d\}$ *be nonempty. Then for a fixed* (α, b, h, β)*, the implication (10) is equivalent to the following system of linear inequalities having a solution* $u \in R^k$*:*

$$B'u + A'\alpha - h = 0,\ -d'u + b - \beta \geq 0, u \geq 0. \tag{11}$$

Proof The implication (10) is equivalent to the following system having no solution $(x, \zeta) \in R^{n+1}$:

$$Bx - d\zeta \leq 0,\ (\alpha'A - h')x + (b - \beta)\zeta < 0,\ -\zeta < 0. \tag{12}$$

By the Motzkin theorem of the alternative (Mangasarian, 1994, Theorem 2.4.2) we have that (12) is equivalent to the following system of inequalities having a solution (u, η, τ):

$$B'u + (A'\alpha - h)\eta = 0,\ -d'u + (b - \beta)\eta - \tau = 0,\ u \geq 0,\ 0 \neq (\eta, \tau) \geq 0. \tag{13}$$

If $\eta = 0$ in (13), then we contradict the nonemptiness of the knowledge set $\{x \mid Bx \leq d\}$. Because, for $x \in \{x \mid Bx \leq d\}$ and (u, τ) that solve (13) with $\eta = 0$, we obtain the contradiction

$$0 \geq u'(Bx - d) = x'B'u - d'u = -d'u = \tau > 0. \tag{14}$$

Hence $\eta > 0$ in (13). Dividing (13) by η and redefining (u, α, τ) as $(\frac{u}{\eta}, \frac{\alpha}{\eta}, \frac{\tau}{\eta})$ we obtain (11). □

Adding the constraints (11) to the linear programming formulation (7) with a linear kernel $K(A, A') = AA'$, we obtain our desired linear program that incorporates the prior knowledge implication (10) into our approximation problem:

$$\begin{array}{cccccc} & & & \min\limits_{(\alpha,b,s,a,u\geq 0)} & e'a + Ce's & \\ \text{s.t.} & -s & \leq & AA'\alpha + be - y & \leq & s, \\ & -a & \leq & \alpha & \leq & a, \\ & & & A'\alpha + B'u & = & h, \\ & & & -d'u & \geq & \beta - b. \end{array} \tag{15}$$

Note that in this linear programming formulation with a linear kernel approximation, both the approximation $w'x + b = \alpha'Ax + b$ to the unknown function f as well as the prior knowledge are linear in the input data A of the problem. This is somewhat restrictive, and therefore we turn now to our principal concern in this work, which is the incorporation of prior knowledge into a *nonlinear* kernel approximation.

3. Knowledge-Based Nonlinear Kernel Approximation

In this part of the paper we will incorporate prior knowledge by using a nonlinear kernel in *both* the linear programming formulation (7) as well as in the prior knowledge implication (10). We begin with the latter, the linear prior knowledge implication (10). If we again consider x as a linear combination of the rows of A, that is

$$x = A't, \tag{16}$$

then the implication (10) becomes

$$Bx \leq d \implies \alpha'AA't + b \geq h'A't + \beta, \tag{17}$$

for a given fixed (α, b). The assumption (16) is not restrictive for the many problems where a sufficiently large number of training data points are available so that any vector in input space can be represented as a linear combination of the training data points.

If we now "kernelize" the various matrix products in the above implication, we have the implication

$$K(B,A')t \leq d \implies \alpha' K(A,A')t + b \geq h'A't + \beta. \tag{18}$$

We note that the two kernels appearing in (18) need not be the same and neither needs to satisfy Mercer's positive semidefiniteness condition. In particular, the first kernel of (18) could be a linear kernel which renders the left side of the implication of (18) the same as that of (17). We note that for a nonlinear kernel, implication (18) is nonlinear in the input space data, but is linear in the implication variable t. We have thus mapped the polyhedral implication (9) into a nonlinear one (18) in the input space data. Assuming for simplicity that the kernel K is symmetric, that is $K(B,A')' = K(A,B')$, it follows directly by Proposition 2.1 that the following equivalence relation holds for implication (18).

Proposition 3.1 Nonlinear Kernel Prior Knowledge Equivalence. *Let the set* $\{t \mid K(B,A')t \leq d\}$ *be nonempty. Then for a given* (α, b, h, β)*, the implication (18) is equivalent to the following system of linear inequalities having a solution* $u \in R^k$*:*

$$K(A,B')u + K(A,A')\alpha - Ah = 0,\ -d'u + b - \beta \geq 0, u \geq 0. \tag{19}$$

We now append the constraints (19), which are equivalent to the nonlinear kernel implication (18), to the linear programming formulation (7). This gives the following linear program for approximating a given function with prior knowledge using a nonlinear kernel:

$$\begin{array}{rcccc} & & \min\limits_{(\alpha,b,s,a,u\geq 0)} & e'a + Ce's & \\ \text{s.t.} & -s \leq & K(A,A')\alpha + be - y & \leq & s, \\ & -a \leq & \alpha & \leq & a, \\ & & K(A,B')u + K(A,A')\alpha & = & Ah, \\ & & -d'u & \geq & \beta - b. \end{array} \tag{20}$$

Since we are not certain that the prior knowledge implication (18) is satisfiable, and since we wish to balance the influence of prior knowledge with that of fitting conventional data points, we need to introduce error variables z and ζ associated with the last two constraints of the linear program (20). These error variables are then driven down by a modified objective function as follows:

$$\begin{array}{rcccc} & & \min\limits_{(\alpha,b,s,a,z,(u,\zeta)\geq 0)} & e'a + Ce's + \mu_1 e'z + \mu_2\zeta & \\ \text{s.t.} & -s \leq & K(A,A')\alpha + be - y & \leq & s, \\ & -a \leq & \alpha & \leq & a, \\ & -z \leq & K(A,B')u + K(A,A')\alpha - Ah & \leq & z, \\ & & -d'u + \zeta & \geq & \beta - b, \end{array} \tag{21}$$

where (μ_1, μ_2) are some positive parameters. This is our final linear program for a single prior knowledge implication. If we have more than one such implication, then the last two sets of constraints are repeated for each implication. For the sake of simplicity we omit these details. The values of the parameters C, μ_1, and μ_2 are chosen so as to balance fitting conventional numerical

data versus the given prior knowledge. One way to choose these parameters is to set aside a "tuning set" of data points and then choose the parameters so as to give a best fit of the tuning set. We also note that all three kernels appearing in (21) could possibly be distinct kernels from each other and none needs to be positive semidefinite. In fact, the kernel $K(A,B')$ could be the linear kernel AB' which was was actually tried in some of our numerical experiments without a noticeable change from using a Gaussian kernel.

We now turn to our numerical experiments.

4. Numerical Experiments

The focus of this paper is mainly theoretical. However, in order to illustrate the power of the proposed formulation, we tested our algorithm on three synthetic examples and one real world example with and without prior knowledge. Two of the synthetic examples are based on the "sinc" function which has been extensively used for kernel approximation testing (Vapnik et al., 1997, Baudat and Anouar, 2001), while the third synthetic example is a two-dimensional hyperboloid. All our results indicate significant improvement due to prior knowledge. The parameters for the synthetic examples were selected using a combination of exhaustive search and a simple variation on the Nelder-Mead simplex algorithm (Nelder and Mead, 1965) that uses only reflection, with average error as the criterion. The chosen parameter values are given in the captions of relevant figures.

4.1 One-Dimensional Sinc Function

We consider the one-dimensional sinc function

$$f(x) = sinc(x) = \frac{sin\pi x}{\pi x}. \tag{22}$$

Given data for the sinc function includes approximate function values for 52 points on the intervals $-3 \leq x \leq -1.4303$ and $1.4303 \leq x \leq 3$. The endpoints ± 1.4303 are approximate local minima of the sinc function. The given approximate function values for $sinc(x)$ are normally perturbed around the true values, with mean 0 and standard deviation 0.5. In addition, there are also three given values at $x = 0$. One of these values is 1, which is the

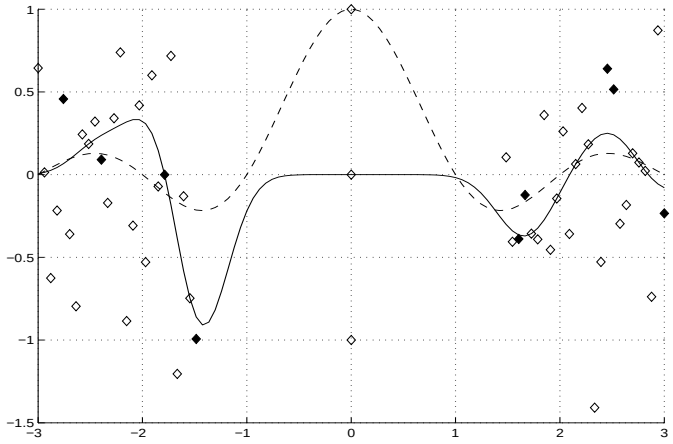


Figure 1: The one-dimensional sinc function $sinc(x) = \frac{sin\pi x}{\pi x}$ (dashed curve) and its Gaussian kernel approximation *without* prior knowledge based on the 55 points shown by diamonds. The nine solid diamonds depict the "support" points used by the nonlinear Gaussian kernel in generating the approximation of $sinc(x)$. That is, they are the rows A_i of A for which $\alpha_i \neq 0$ in the solution of the nonlinear Gaussian kernel approximation of (7) for $f(x)$: $f(x) \approx K(x', A')\alpha + b$. The approximation has an average error of 0.3113 over a grid of 100 points in the interval $[-3,3]$. Parameter values used: $\mu = 7, C = 5$.

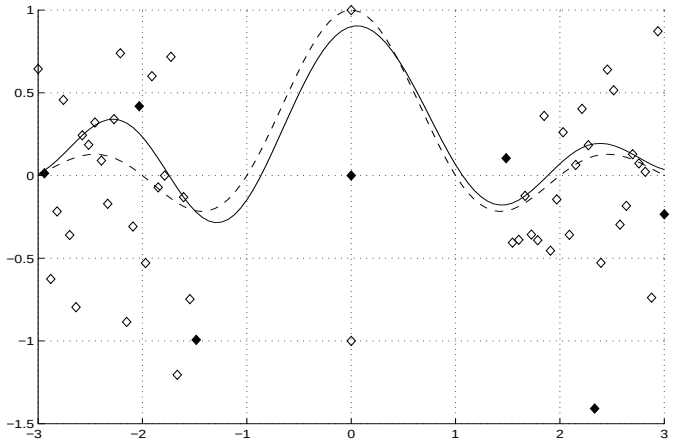


Figure 2: The one-dimensional sinc function $sinc(x) = \frac{sin\pi x}{\pi x}$ (dashed curve) and its Gaussian kernel approximation *with* prior knowledge based on 55 points, shown by diamonds, which are the same as those of Figure 1. The seven solid diamonds depict the "support" points used by the nonlinear Gaussian kernel in generating the approximation of $sinc(x)$. The prior knowledge consists of the implication $-\frac{1}{4} \leq x \leq \frac{1}{4} \Rightarrow f(x) \geq \frac{\sin(\pi/4)}{\pi/4}$, which is implemented by replacing $f(x)$ by its nonlinear kernel approximation (23). The approximation has an average error of 0.0901 over a grid of 100 points in the interval $[-3,3]$, which is less than $\frac{1}{3.4}$ times the error of Figure 1. Parameter values used: $\mu = 1, C = 13, \mu_1 = 5, \mu_2 = 450$.

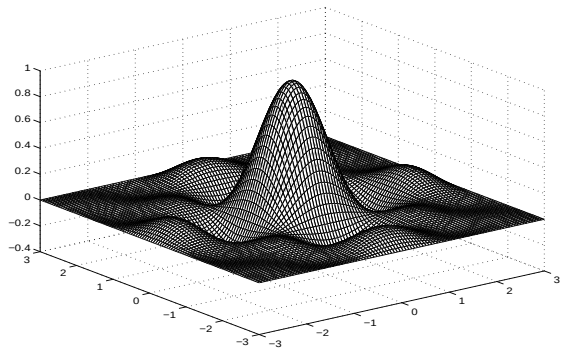


Figure 3: The exact product sinc function $f(x_1,x_2) = \frac{sin\pi x_1}{\pi x_1}\frac{sin\pi x_2}{\pi x_2}$.

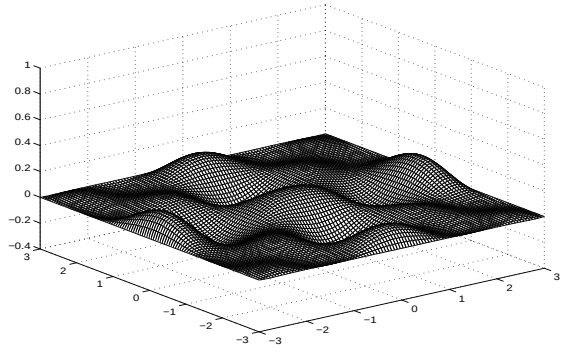


Figure 4: Gaussian kernel approximation of the product sinc function $f(x_1,x_2) = \frac{sin\pi x_1}{\pi x_1}\frac{sin\pi x_2}{\pi x_2}$ based on 211 exact function values plus 2 incorrect function values, but *without* prior knowledge. The approximation has an average error of 0.0501 over a grid of 2500 points in the set $\{[-3,3]\times[-3,3]\}$. Parameter values used: $\mu = 0.2, C = 10^6$.

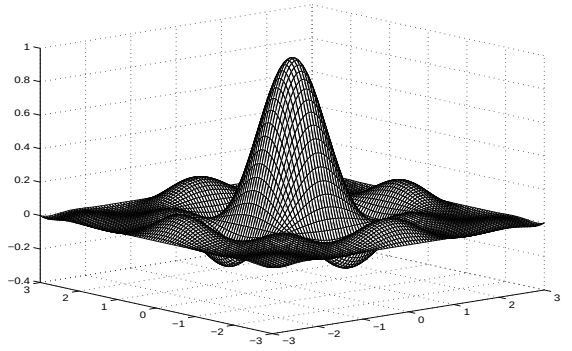


Figure 5: Gaussian kernel approximation of the product sinc function based on the same 213 function values as Figure 4 *plus* prior knowledge consisting of $(x_1,x_2) \in \{[-0.1,0.1]\times[-0.1,0.1]\} \Rightarrow f(x_1,x_2) \geq (\frac{sin(\pi/10)}{\pi/10})^2$. The approximation has an average error of 0.0045 over a grid of 2500 points in the set $\{[-3,3]\times[-3,3]\}$, which is less than $\frac{1}{11.1}$ times the error of Figure 4. Parameters are $\mu = 1, C = 16000, \mu_1 = 15000, \mu_2 = 5\cdot 10^6$.

actual limit of the sinc function at 0. The other values at $x = 0$ are 0 and -1 which are intended to be misleading to the approximation.

Figure 1 depicts $sinc(x)$ by a dashed curve and its approximation *without* prior knowledge by a solid curve based on the 55 points shown by diamonds. The nine solid diamonds depict "support" points, that is rows A_i of A for which $\alpha_i \neq 0$ in the solution of the nonlinear Gaussian kernel approximation of (7) for $f(x)$:

$$f(x) \approx K(x', A')\alpha + b. \tag{23}$$

The approximation in Figure 1 has an average error of 0.3113. This error is computed by averaging over a grid of 100 equally spaced points in the interval $[-3, 3]$.

Figure 2 depicts $sinc(x)$ by a dashed curve and its much better approximation *with* prior knowledge by a solid curve based on the 55 points shown, which are the same as those of Figure 1. The seven solid diamond points are "support" points, that is rows A_i of A for which $\alpha_i \neq 0$ in the solution of the nonlinear Gaussian kernel approximation (23) of (21) for $f(x)$. The approximation in Figure 2 has an average error of 0.0901 computed over a grid of 100 equally spaced points on $[-3, 3]$. The prior knowledge used to approximate the one-dimensional sinc function is $-\frac{1}{4} \leq x \leq \frac{1}{4} \Rightarrow f(x) \geq \frac{sin(\pi/4)}{\pi/4}$. The value $\frac{sin(\pi/4)}{\pi/4}$ is the minimum of $sinc(x)$ on the knowledge interval $[-\frac{1}{4}, \frac{1}{4}]$. This prior knowledge is implemented by replacing $f(x)$ by its nonlinear kernel approximation (23) and then using the implication (18) as follows:

$$K(I, A')t \leq \frac{1}{4} \wedge K(-I, A')t \leq \frac{1}{4} \Longrightarrow \alpha' K(A, A')t + b \geq \frac{sin(\pi/4)}{\pi/4}. \tag{24}$$

4.2 Two-Dimensional Sinc Function

Our second example is the two-dimensional $sinc(x)$ function for $x \in R^2$:

$$f(x_1, x_2) = sinc(x_1)sinc(x_2) = \frac{sin\pi x_1}{\pi x_1} \frac{sin\pi x_2}{\pi x_2}. \tag{25}$$

The given data for the two-dimensional sinc function includes 210 points in the region $\{(x_1, x_2) | (-3 \leq x_1 \leq -1.4303 \vee 1.4303 \leq x_1 \leq 3) \wedge (-3 \leq x_2 \leq -1 \vee 1 \leq x_2 \leq 3)\}$. This region excludes the largest bump in the function centered at $(x_1, x_2) = (0, 0)$. The given values are exact function values. There are also three values given at $(x_1, x_2) = (0, 0)$, similar to the previous example with the one dimensional sinc. The first value is the actual limit of the function at $(0, 0)$, which is 1. The other two values are 0 and -1. These last two values are intended to mislead the approximation.

Figure 3 depicts the two-dimensional sinc function of (25). Figure 4 depicts an approximation of $sinc(x_1)sinc(x_2)$ *without* prior knowledge by a surface based on the 213 points described above. The approximation in Figure 4 has an average error of 0.0501. This value is computed by averaging over a grid of 2500 equally spaced points in $\{[-3, 3] \times [-3, 3]\}$.

Figure 5 depicts a much better approximation of $sinc(x_1)sinc(x_2)$ *with* prior knowledge by a surface based on the same 213 points. The approximation in Figure 5 has an average error of 0.0045. This value is computed by averaging over 2500 equally spaced points in

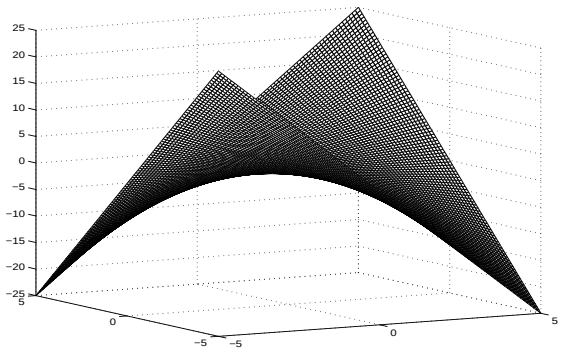


Figure 6: The exact hyperboloid function $f(x_1,x_2) = x_1x_2$.

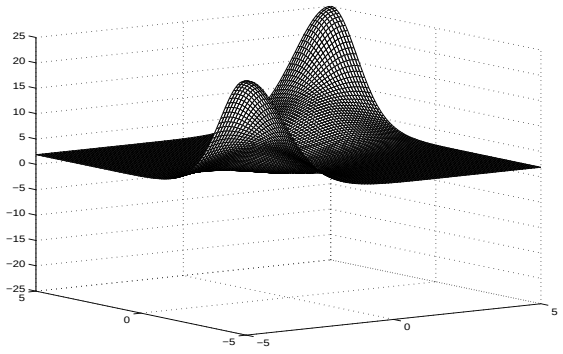


Figure 7: Gaussian kernel approximation of the hyperboloid function $f(x_1,x_2) = x_1x_2$ based on 11 exact function values along the line $x_2 = x_1, x_1 \in \{-5,-4,\ldots,4,5\}$, but *without* prior knowledge. The approximation has an average error of 4.8351 over 2500 points in the set $\{[-5,5]\times[-5,5]\}$. Parameter values used: $\mu = 0.361, C = 145110$.

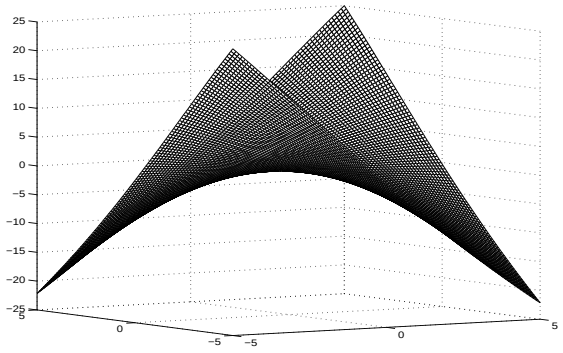


Figure 8: Gaussian kernel approximation of the hyperboloid function $f(x_1,x_2) = x_1x_2$ based on the same 11 function values as of Figure 7 *plus* prior knowledge consisting of the implications (27) and (28). The approximation has an average error of 0.2023 over 2500 points in the set $\{[-5,5]\times[-5,5]\}$, which is less than $\frac{1}{23.9}$ times the error of Figure 7. Parameter values used: $\mu = 0.0052, C = 5356, \mu_1 = 685, \mu_2 = 670613$.

$\{[-3,3]\times[-3,3]\}$. The prior knowledge consists of the implication

$$(x_1,x_2)\in\{[-0.1,0.1]\times[-0.1,0.1]\}\Rightarrow f(x_1,x_2)\geq(\frac{sin(\pi/10)}{\pi/10})^2.$$

The value $(\frac{sin(\pi/10)}{\pi/10})^2$ is equal to the minimum value of $sinc(x_1)sinc(x_2)$ on the knowledge set $\{[-0.1,0.1]\times[-0.1,0.1]\}$. This prior knowledge is implemented by replacing $f(x_1,x_2)$ by its nonlinear kernel approximation (23) and then using the implication (18).

4.3 Two-Dimensional Hyperboloid Function

Our third example is the two-dimensional hyperboloid function

$$f(x_1,x_2)=x_1x_2. \tag{26}$$

For the two-dimensional hyperboloid function, the given data consists of 11 points along the line $x_2=x_1, x_1\in\{-5,-4,\ldots,4,5\}$. The given values at these points are the actual function values.

Figure 6 depicts the two-dimensional hyperboloid function of (26). Figure 7 depicts an approximation of the hyperboloid function, *without* prior knowledge, by a surface based on the 11 points described above. The approximation in Figure 7 has an average error of 4.8351 computed over a grid of 2500 equally spaced points in $\{[-5,5]\times[-5,5]\}$.

Figure 8 depicts a much better approximation of the hyperboloid function by a nonlinear surface based on the same 11 points above *plus* prior knowledge. The approximation in Figure 8 has an average error of 0.2023 computed over a grid of 2500 equally spaced points in $\{[-5,5]\times[-5,5]\}$. The prior knowledge consists of the following two implications:

$$(x_1,x_2)\in\{(x_1,x_2)|-\frac{1}{3}x_1\leq x_2\leq-\frac{2}{3}x_1\}\Rightarrow f(x_1,x_2)\leq 10x_1 \tag{27}$$

and

$$(x_1,x_2)\in\{(x_1,x_2)|-\frac{2}{3}x_1\leq x_2\leq-\frac{1}{3}x_1\}\Rightarrow f(x_1,x_2)\leq 10x_2. \tag{28}$$

These implications are implemented by replacing $f(x_1,x_2)$ by its nonlinear kernel approximation (23) and then using the implication (18). The regions on which the knowledge is given are cones on which x_1x_2 is negative. Since the two implications are analogous, we explain (27) only. This implication is justified on the basis that $x_1x_2\leq 10x_1$ over the knowledge cone $\{(x_1,x_2)|-\frac{1}{3}x_1\leq x_2\leq-\frac{2}{3}x_1\}$ for sufficiently large x_2, that is $x_2\geq 10$. This is intended to capture coarsely the global shape of $f(x_1,x_2)$ and succeeds in generating a more accurate overall approximation of the function.

4.4 Predicting Lymph Node Metastasis

We conclude our numerical results with a potentially useful application of knowledge-based approximation to breast cancer prognosis (Mangasarian et al., 1995, Wolberg et al., 1995, Lee et al., 2001). An important prognostic indicator for breast cancer recurrence is the number of metastasized lymph nodes under a patient's armpit, which could be as many as 30. To determine this number, a patient must undergo optional surgery in addition to the removal of the breast tumor. If the predicted number of metastasized lymph nodes is sufficiently small, then the oncological surgeon may decide not to perform the additional surgery. Thus, it is useful to approximate the number of metastasized

lymph nodes as a function of thirty available cytological features and one histological feature. The cytological features are obtained from a fine needle aspirate during the diagnostic procedure while the histological feature is obtained during surgery. Our proposed knowledge-based approximation can be used to improve the determination of such a function, $f : R^{31} \longrightarrow R$, that predicts the number of metastasized lymph nodes. For example, in certain polyhedral regions of R^{31}, past training data indicate the existence of a substantial number of metastasized lymph nodes, whereas certain other regions indicate the unlikely presence of any metastasis. This knowledge can be applied to obtain a hopefully more accurate lymph node function f than that based on numerical function approximation alone.

We have performed preliminary experiments with the Wisconsin Prognostic Breast Cancer (WPBC) data available from (Murphy and Aha, 1992). In our experiments we reduced R^{31} to R^4 and predicted the number of metastasized lymph nodes based on three cytological features: mean cell texture, worst cell smoothness, and worst cell area, as well as the histological feature tumor size. The tumor size is an obvious histological feature to include, while the three other cytological features were the same as those selected for breast cancer diagnosis in (Mangasarian, 2001). Thus, we are approximating a function $f : R^4 \longrightarrow R$. Note that the online version of the WPBC data contains four entries with no lymph node information which were removed for our experiments. After removing these entries, we were left with 194 examples in our dataset.

To simulate the procedure of an expert obtaining prior knowledge from past data we used the following procedure. First we took a random 20% of the dataset to analyze as "past data". Inspecting this past data, we choose the following background knowledge:

$$x_1 \geq 22.4 \wedge x_2 \geq 0.1 \wedge x_3 \geq 1458.9 \wedge x_4 \geq 3.1 \implies f(x_1, x_2, x_3, x_4) \geq 1, \tag{29}$$

where x_1, x_2, x_3, and x_4 denote mean texture, worst smoothness, worst area, and tumor size respectively. This prior knowledge is based on a typical oncological surgeon's advice that larger values of the variables are likely to result in more metastasized lymph nodes. The constants in (29) were chosen by taking the average values of $x_1, \ldots, x_4$ for the entries in the past data with at least one metastasized lymph node.

We used ten-fold cross validation to compare the average absolute error between an approximation without prior knowledge and an approximation with the prior knowledge of Equation (29) on the 80% of the data that was not used as "past data" to generate the constants in (29). Parameters in (21) using a Gaussian kernel were chosen using the Nelder-Mead algorithm on a tuning set taken from the training data for each fold. The average absolute error of the function approximation with no prior knowledge was 3.75 while the average absolute error with prior knowledge was 3.35, a 10.5% reduction. The mean function value of the data used in the ten-fold cross validation experiments is 3.30, so neither approximation is accurate. However, these results indicate that adding prior knowledge does indeed improve the function approximation substantially. Hopefully more sophisticated prior knowledge, based on a more detailed analysis of the data and consultation with domain experts, will further reduce the error.

We close this section with a potential application to a reinforcement learning task (Sutton and Barto, 1998), where the goal is to predict the value of taking an action at a given state. Thus, the domain of the function to be approximated is the Cartesian product of the set of states and the set of actions. In particular, we plan to use the *Keep-Away* subtask of the soccer game developed in (Stone and Sutton, 2001). The state description includes measurements such as distance to each of the opposing players, distance to the soccer ball, distances to the edges of the field, etc. Actions include

holding the ball and attempting a pass to a teammate. It has been demonstrated that providing prior knowledge can improve the choice of actions significantly (Kuhlmann et al., 2004, Maclin and Shavlik, 1996). One example of advice (that is, prior knowledge) that has been successfully used in this domain is the simple advice that "if no opponent is within 8 meters, holding the ball is a good idea." In our approach we approximate a value function v as a function of states and actions. Advice can be stated as the following implication, assuming two opponents:

$$d_1 \geq 8 \wedge d_2 \geq 8 \wedge a = h \implies v \geq c, \tag{30}$$

where d_1 denotes the distance to Opponent 1, d_2 the distance to Opponent 2, $a = h$ the action of holding the ball, v the predicted value, and c is some constant. It is hoped that this "advice" can help in generating an improved value function v based on the current description of the state of the soccer game.

5. Conclusion and Outlook

We have presented a knowledge-based formulation of a nonlinear kernel SVM approximation. The approximation is obtained using a linear programming formulation with any nonlinear symmetric kernel and with no positive semidefiniteness (Mercer) condition assumed. The issues associated with sampling the knowledge sets in order to generate function values (that is, a matrix A and a corresponding vector y) in situations where there are no conventional data points constitute an interesting topic for future research. Additional future work includes refinement of prior knowledge and applications to medical problems, computer vision, microarray gene classification, and efficacy of drug treatment, all of which have prior knowledge available.

Acknowledgments

We are grateful to our colleagues Rich Maclin and Dave Musicant for constructive comments. Research described in this UW Data Mining Institute Report 03-05, October 2003, was supported by NSF Grants CCR-0138308, and IRI-9502990, by NLM Grant 1 R01 LM07050-01, by DARPA ISTO Grant HR0011-04-0007, by PHS Grant 5 T15 LM07359-02 and by Microsoft.

References

G. Baudat and F. Anouar. Kernel-based methods and function approximation. In *International Joint Conference on Neural Networks*, pages 1244–1249, Washington, D.C., 2001.

V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods*. John Wiley & Sons, New York, 1998.

F. Deutsch. *Best Approximation in Inner Product Spaces*. Springer-Verlag, Berlin, 2001.

H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems -9-*, pages 155–161, Cambridge, MA, 1997. MIT Press.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 171–203, Cambridge, MA, 2000. MIT Press.

G. Fung, O. L. Mangasarian, and J. Shavlik. Knowledge-based nonlinear kernel classifiers. Technical Report 03-02, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, March 2003a. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-03.ps. *Conference on Learning Theory (COLT 03) and Workshop on Kernel Machines*, Washington D.C., August 24-27, 2003. Proceedings edited by M. Warmuth and B. Schölkopf, Springer Verlag, Berlin, 2003, 102-113.

G. Fung, O. L. Mangasarian, and J. Shavlik. Knowledge-based support vector machine classifiers. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 521–528. MIT Press, Cambridge, MA, October 2003b. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-09.ps.

G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *Proceedings of the AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*, San Jose, CA, 2004.

Y.-J. Lee, O. L. Mangasarian, and W. H. Wolberg. Survival-time classification of breast cancer patients. Technical Report 01-03, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, March 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-03.ps. *Computational Optimization and Applications* 25, 2003, 151-166.

R. Maclin and J. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22, 1996.

O. L. Mangasarian. *Nonlinear Programming*. SIAM, Philadelphia, PA, 1994.

O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.

O. L. Mangasarian. Data mining via support vector machines, July 23-27, 2001. http://ftp.cs.wisc.edu/math-prog/talks/ifip3tt.ppt.

O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46:255–269, 2002. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/99-02.ps.

O. L. Mangasarian and L. L. Schumaker. Splines via optimal control. In I. J. Schoenberg, editor, *Approximations with Special Emphasis on Splines*, pages 119–156, New York, 1969. Academic Press.

O. L. Mangasarian and L. L. Schumaker. Discrete splines via mathematical programming. *SIAM Journal on Control*, 9:174–183, May 1971.

O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.

C. A. Micchelli and F. I. Utreras. Smoothing and interpolation in a convex subset of a hilbert space. *SIAM Journal of Statistical Computing*, 9:728–746, 1988.

P. M. Murphy and D. W. Aha. UCI machine learning repository, 1992. www.ics.uci.edu/∼mlearn/MLRepository.html.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7: 308–313, 1965.

B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640 – 646, Cambridge, MA, 1998. MIT Press.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

A. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211–231, 1998.

P. Stone and R. Sutton. Scaling reinforcement learning toward robocup soccer. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01)*, Williams, MA, 2001.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, second edition, 2000.

V. N. Vapnik, S. E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation and signal processing. In *Neural Information Processing Systems Volume 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.

W. H. Wolberg, W. N. Street, D. N. Heisey, and O. L. Mangasarian. Computerized breast cancer diagnosis and prognosis from fine-needle aspirates. *Archives of Surgery*, 130:511–516, 1995.

Journal of Machine Learning Research 5 (2004) 1143–1175 Submitted 9/03; Revised 4/04; Published 9/04

Support Vector Machine Soft Margin Classifiers: Error Analysis

Di-Rong Chen DRCHEN@BUAA.EDU.CN
Department of Applied Mathematics
Beijing University of Aeronautics and Astronautics
Beijing 100083, P. R. CHINA

Qiang Wu WU.QIANG@STUDENT.CITYU.EDU.HK
Yiming Ying YMYING@CITYU.EDU.HK
Ding-Xuan Zhou MAZHOU@MATH.CITYU.EDU.HK
Department of Mathematics
City University of Hong Kong
Kowloon, Hong Kong, P. R. CHINA

Editor: Peter Bartlett

Abstract

The purpose of this paper is to provide a PAC error analysis for the q-norm soft margin classifier, a support vector machine classification algorithm. It consists of two parts: regularization error and sample error. While many techniques are available for treating the sample error, much less is known for the regularization error and the corresponding approximation error for reproducing kernel Hilbert spaces. We are mainly concerned about the regularization error. It is estimated for general distributions by a K-functional in weighted L^q spaces. For weakly separable distributions (i.e., the margin may be zero) satisfactory convergence rates are provided by means of separating functions. A projection operator is introduced, which leads to better sample error estimates especially for small complexity kernels. The misclassification error is bounded by the V-risk associated with a general class of loss functions V. The difficulty of bounding the offset is overcome. Polynomial kernels and Gaussian kernels are used to demonstrate the main results. The choice of the regularization parameter plays an important role in our analysis.

Keywords: support vector machine classification, misclassification error, q-norm soft margin classifier, regularization error, approximation error

1. Introduction

In this paper we study support vector machine (SVM) classification algorithms and investigate the SVM q-norm soft margin classifier with $1 < q < \infty$. Our purpose is to provide an error analysis for this algorithm in the PAC framework.

Let (X,d) be a compact metric space and $Y = \{1,-1\}$. A binary classifier $f : X \to \{1,-1\}$ is a function from X to Y which divides the input space X into two classes.

Let ρ be a probability distribution on $Z := X \times Y$ and $(\mathcal{X},\mathcal{Y})$ be the corresponding random variable. The *misclassification error* for a classifier $f : X \to Y$ is defined to be the probability of the event $\{f(\mathcal{X}) \neq \mathcal{Y}\}$:

$$\mathcal{R}(f) := \text{Prob}\{f(\mathcal{X}) \neq \mathcal{Y}\} = \int_X P(\mathcal{Y} \neq f(x)|x) d\rho_X(x). \tag{1}$$

Here ρ_X is the marginal distribution on X and $P(\cdot|x)$ is the conditional probability measure given $\mathcal{X} = x$.

The SVM q-norm soft margin classifier (Cortes and Vapnik, 1995; Vapnik, 1998) is constructed from samples and depends on a reproducing kernel Hilbert space associated with a Mercer kernel.

Let $K : X \times X \to \mathbb{R}$ be continuous, symmetric and positive semidefinite, i.e., for any finite set of distinct points $\{x_1, \cdots, x_\ell\} \subset X$, the matrix $(K(x_i, x_j))_{i,j=1}^{\ell}$ is positive semidefinite. Such a kernel is called a *Mercer kernel*.

The *Reproducing Kernel Hilbert Space* (RKHS) $\mathcal{H}_K$ associated with the kernel K is defined (Aronszajn, 1950) to be the closure of the linear span of the set of functions $\{K_x := K(x, \cdot) : x \in X\}$ with the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_K} = \langle \cdot, \cdot \rangle_K$ satisfying $\langle K_x, K_y \rangle_K = K(x, y)$ and

$$\langle K_x, g \rangle_K = g(x), \qquad \forall x \in X, g \in \mathcal{H}_K.$$

Denote $C(X)$ as the space of continuous functions on X with the norm $\| \cdot \|_\infty$. Let $\kappa := \sqrt{\|K\|_\infty}$. Then the above reproducing property tells us that

$$\|g\|_\infty \le \kappa \|g\|_K, \qquad \forall g \in \mathcal{H}_K. \tag{2}$$

Define $\overline{\mathcal{H}_K} := \mathcal{H}_K + \mathbb{R}$. For a function $f = f_1 + b$ with $f_1 \in \mathcal{H}_K$ and $b \in \mathbb{R}$, we denote $f^* = f_1$ and $b_f = b \in \mathbb{R}$. The constant term b is called the *offset*. For a function $f : X \to \mathbb{R}$, the sign function is defined as $\operatorname{sgn}(f)(x) = 1$ if $f(x) \ge 0$ and $\operatorname{sgn}(f)(x) = -1$ if $f(x) < 0$.

Now the *SVM q-norm soft margin classifier* (SVM q-classifier) associated with the Mercer kernel K is defined as $\operatorname{sgn}(f_{\mathbf{z}})$, where $f_{\mathbf{z}}$ is a minimizer of the following optimization problem involving a set of random samples $\mathbf{z} = (x_i, y_i)_{i=1}^m \in Z^m$ independently drawn according to ρ:

$$\begin{aligned} f_{\mathbf{z}} := \arg \min_{f \in \overline{\mathcal{H}_K}} \quad & \frac{1}{2} \|f^*\|_K^2 + \frac{C}{m} \sum_{i=1}^m \xi_i^q, \\ \text{subject to} \quad & y_i f(x_i) \ge 1 - \xi_i, \text{ and } \xi_i \ge 0 \text{ for } i = 1, \ldots, m. \end{aligned} \tag{3}$$

Here C is a constant which depends on m: $C = C(m)$, and often $\lim_{m \to \infty} C(m) = \infty$.

Throughout the paper, we assume $1 < q < \infty$, $m \in \mathbb{N}$, $C > 0$, and $\mathbf{z} = (x_i, y_i)_{i=1}^m$ are random samples independently drawn according to ρ. Our target is to understand how $\operatorname{sgn}(f_{\mathbf{z}})$ converges (with respect to the misclassification error) to the best classifier, the Bayes rule, as m and hence $C(m)$ tend to infinity. Recall the regression function of ρ:

$$f_\rho(x) = \int_Y y d\rho(y|x) = P(\mathcal{Y} = 1|x) - P(\mathcal{Y} = -1|x), \qquad x \in X. \tag{4}$$

Then the *Bayes rule* is given (e.g. Devroye, L. Györfi and G. Lugosi, 1997) by the sign of the regression function $f_c := \operatorname{sgn}(f_\rho)$. Estimating the excess misclassification error

$$\mathcal{R}(\operatorname{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c) \tag{5}$$

for the classification algorithm (3) is our goal. In particular, we try to understand how the choice of the regularization parameter C affects the error.

To investigate the error bounds for general kernels, we rewrite (3) as a regularization scheme. Define the loss function $V = V_q$ as

$$V(y, f(x)) := (1 - yf(x))_+^q = |y - f(x)|^q \chi_{\{yf(x) \le 1\}}, \tag{6}$$

where $(t)_+ = \max\{0,t\}$. The corresponding *V-risk* is

$$\mathcal{E}(f) := E(V(y,f(x))) = \int_Z V(y,f(x))d\rho(x,y). \tag{7}$$

If we set the empirical error as

$$\mathcal{E}_{\mathbf{z}}(f) := \frac{1}{m}\sum_{i=1}^m V(y_i,f(x_i)) = \frac{1}{m}\sum_{i=1}^m (1-y_i f(x_i))_+^q, \tag{8}$$

then the scheme (3) can be written as (see Evgeniou, Pontil and Poggio, 2000)

$$f_{\mathbf{z}} = \arg\min_{f\in\overline{\mathcal{H}}_K}\left\{\mathcal{E}_{\mathbf{z}}(f) + \frac{1}{2C}\|f^*\|_K^2\right\}. \tag{9}$$

Notice that when $\overline{\mathcal{H}}_K$ is replaced by $\mathcal{H}_K$, the scheme (9) is exactly the Tikhonov regularization scheme (Tikhonov and Arsenin, 1977) associated with the loss function V. So one may hope that the method for analyzing regularization schemes can be applied.

The definitions of the V-risk (7) and the empirical error (8) tell us that for a function $f = f^* + b \in \overline{\mathcal{H}}_K$, the random variable $\xi = V(y,f(x))$ on Z has the mean $\mathcal{E}(f)$ and $\frac{1}{m}\sum_{i=1}^m \xi(z_i) = \mathcal{E}_{\mathbf{z}}(f)$. Thus we may expect by some standard empirical risk minimization (ERM) argument (e.g. Cucker and Smale, 2001; Evgeniou, Pontil and Poggio, 2000; Shawe-Taylor et al., 1998; Vapnik, 1998; Wahba, 1990) to derive bounds for $\mathcal{E}(f_{\mathbf{z}}) - \mathcal{E}(f_q)$, where f_q is a minimizer of the V-risk (7). It was shown in Lin (2002) that for $q > 1$ such a minimizer is given by

$$f_q(x) = \frac{(1+f_\rho(x))^{1/(q-1)} - (1-f_\rho(x))^{1/(q-1)}}{(1+f_\rho(x))^{1/(q-1)} + (1-f_\rho(x))^{1/(q-1)}}, \qquad x \in X. \tag{10}$$

For $q = 1$ a minimizer is f_c, see Wahba (1999). Note that $\operatorname{sgn}(f_q) = f_c$.

Recall that for the classification algorithm (3), we are interested in the excess misclassification error (5), not the excess V-risk $\mathcal{E}(f_{\mathbf{z}}) - \mathcal{E}(f_q)$. But we shall see in Section 3 that $\mathcal{R}(\operatorname{sgn}(f)) - \mathcal{R}(f_c) \le \sqrt{2(\mathcal{E}(f) - \mathcal{E}(f_q))}$. One might apply this to the function $f_{\mathbf{z}}$ and get estimates for (5). However, special features of the loss function (6) enable us to do better: by restricting $f_{\mathbf{z}}$ onto $[-1,1]$, we can improve the sample error estimates. The idea of the following projection operator was introduced for this purpose in Bartlett (1998).

Definition 1 *The* projection operator π *is defined on the space of measurable functions* $f : X \to \mathbb{R}$ *as*

$$\pi(f)(x) = \begin{cases} 1, & \text{if } f(x) \ge 1, \\ -1, & \text{if } f(x) \le -1, \\ f(x), & \text{if } -1 < f(x) < 1. \end{cases} \tag{11}$$

It is trivial that $\operatorname{sgn}(\pi(f)) = \operatorname{sgn}(f)$. Hence

$$\mathcal{R}(\operatorname{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c) \le \sqrt{2(\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q))}. \tag{12}$$

The definition of the loss function (6) also tells us that $V(y,\pi(f)(x)) \le V(y,f(x))$, so

$$\mathcal{E}(\pi(f)) \le \mathcal{E}(f) \qquad \text{and} \qquad \mathcal{E}_{\mathbf{z}}(\pi(f)) \le \mathcal{E}_{\mathbf{z}}(f). \tag{13}$$

According to (12), we need to estimate $\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q)$ in order to bound (5). To this end, we introduce a *regularizing function* $f_{K,C} \in \overline{\mathcal{H}}_K$. It is arbitrarily chosen and depends on C.

Proposition 2 *Let $f_{K,C} \in \overline{\mathcal{H}_K}$, and $f_{\mathbf{z}}$ be defined by (9). Then $\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q)$ can be bounded by*

$$\left\{\mathcal{E}(f_{K,C}) - \mathcal{E}(f_q) + \frac{1}{2C}\|f^*_{K,C}\|^2_K\right\} + \left\{\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}_{\mathbf{z}}(\pi(f_{\mathbf{z}})) + \mathcal{E}_{\mathbf{z}}(f_{K,C}) - \mathcal{E}(f_{K,C})\right\}. \tag{14}$$

Proof Decompose the difference $\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q)$ as

$$\begin{aligned}&\left\{\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}_{\mathbf{z}}(\pi(f_{\mathbf{z}}))\right\} + \left\{\left(\mathcal{E}_{\mathbf{z}}(\pi(f_{\mathbf{z}})) + \frac{1}{2C}\|f^*_{\mathbf{z}}\|^2_K\right) - \left(\mathcal{E}_{\mathbf{z}}(f_{K,C}) + \frac{1}{2C}\|f^*_{K,C}\|^2_K\right)\right\}\\ &\quad + \left\{\mathcal{E}_{\mathbf{z}}(f_{K,C}) - \mathcal{E}(f_{K,C})\right\} + \left\{\mathcal{E}(f_{K,C}) - \mathcal{E}(f_q) + \frac{1}{2C}\|f^*_{K,C}\|^2_K\right\} - \frac{1}{2C}\|f^*_{\mathbf{z}}\|^2_K.\end{aligned}$$

By the definition of $f_{\mathbf{z}}$ and (13), the second term is ≤ 0. Then the statement follows. ■

The first term in (14) is called the regularization error (Smale and Zhou, 2004). It can be expressed as a generalized K-functional $\inf_{f\in\overline{\mathcal{H}_K}}\left\{\mathcal{E}(f) - \mathcal{E}(f_q) + \frac{1}{2C}\|f^*\|^2_K\right\}$ when $f_{K,C}$ takes a special choice $\widetilde{f}_{K,C}$ (a standard choice in the literature, e.g. Steinwart 2001) defined as

$$\widetilde{f}_{K,C} := \arg\min_{f\in\overline{\mathcal{H}_K}}\left\{\mathcal{E}(f) + \frac{1}{2C}\|f^*\|^2_K\right\}. \tag{15}$$

Definition 3 *Let V be a general loss function and f^V_ρ be a minimizer of the V-risk (7). The* regularization error *for the regularizing function $f_{K,C} \in \overline{\mathcal{H}_K}$ is defined as*

$$\mathcal{D}(C) := \mathcal{E}(f_{K,C}) - \mathcal{E}(f^V_\rho) + \frac{1}{2C}\|f^*_{K,C}\|^2_K. \tag{16}$$

It is called the regularization error of the scheme *(9) when $f_{K,C} = \widetilde{f}_{K,C}$:*

$$\widetilde{\mathcal{D}}(C) := \inf_{f\in\overline{\mathcal{H}_K}}\left\{\mathcal{E}(f) - \mathcal{E}(f^V_\rho) + \frac{1}{2C}\|f^*\|^2_K\right\}.$$

The main concern of this paper is the regularization error. We shall investigate its asymptotic behavior. This investigation is not only important for bounding the first term in (14), but also crucial for bounding the second term (sample error): it is well known in structural risk minimization (Shawe-Taylor et al., 1998) that the size of the hypothesis space is essential. This is determined by $\mathcal{D}(C)$ in our setting. Once C is fixed, the sample error estimate becomes routine. Therefore, we need to understand the choice of the parameter C from the bound for $\mathcal{D}(C)$.

Proposition 4 *For any $C \ge 1/2$ and any $f_{K,C} \in \overline{\mathcal{H}_K}$, there holds*

$$\mathcal{D}(C) \ge \widetilde{\mathcal{D}}(C) \ge \frac{\widetilde{\kappa}^2}{2C} \tag{17}$$

where

$$\widetilde{\kappa} := \mathcal{E}_0/(1 + \kappa q 4^{q-1}), \qquad \mathcal{E}_0 := \inf_{b\in\mathbb{R}}\{\mathcal{E}(b) - \mathcal{E}(f_q)\}. \tag{18}$$

Moreover, $\widetilde{\kappa} = 0$ if and only if for some $p_0 \in [0,1]$, $P(\mathcal{Y} = 1|x) = p_0$ in probability.

This proposition will be proved in Section 5.

According to Proposition 4, the decay of $\mathcal{D}(C)$ cannot be faster than $O(1/C)$ except for some very special distributions. This special case is caused by the offset in (9), for which $\widetilde{\mathcal{D}}(C) \equiv 0$. Throughout this paper we shall ignore this trivial case and assume $\widetilde{\kappa} > 0$.

When ρ is strictly separable, $\mathcal{D}(C) = O(1/C)$. But this is a very special phenomenon. In general, one should not expect $\mathcal{E}(f) = \mathcal{E}(f_q)$ for some $f \in \overline{\mathcal{H}_K}$. Even for (weakly) separable distributions with zero margin, $\mathcal{D}(C)$ decays as $O(C^{-p})$ for some $0 < p < 1$. To realize such a decay for these separable distributions, the regularizing function will be multiples of a separating function. For details and the concepts of strictly or weakly separable distributions, see Section 2.

For general distributions, we shall choose $f_{K,C} = \widetilde{f}_{K,C}$ in Sections 6 and 7 and estimate the regularization error of the scheme (9) associated with the loss function (6) by means of the approximation in the function space $L^q_{\rho_X}$. In particular, $\widetilde{\mathcal{D}}(C) \le \mathcal{K}(f_q, \frac{1}{2C})$, where $\mathcal{K}(f_q, t)$ is a K-functional defined as

$$\mathcal{K}(f_q,t) := \begin{cases} \inf\limits_{f\in\overline{\mathcal{H}_K}} \left\{ \|f - f_q\|^q_{L^q_{\rho_X}} + t\|f^*\|^2_K \right\}, & \text{if } 1 < q \le 2, \\ \inf\limits_{f\in\overline{\mathcal{H}_K}} \left\{ q2^{q-1}(2^{q-1}+1)\|f - f_q\|_{L^q_{\rho_X}} + t\|f^*\|^2_K \right\}, & \text{if } q > 2. \end{cases} \tag{19}$$

In the case $q = 1$, the regularization error (Wu and Zhou, 2004) depends on the approximation in $L^1_{\rho_X}$ of the function f_c which is not continuous in general. For $q > 1$, the regularization error depends on the approximation in $L^q_{\rho_X}$ of the function f_q. When the regression function has good smoothness, f_q has much higher regularity than f_c. Hence the convergence for $q > 1$ may be faster than that for $q = 1$, which improves the regularization error.

The second term in (14) is called the *sample error*. When C is fixed, the sample error $\mathcal{E}(f_{\mathbf{z}}) - \mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}})$ is well understood (except for the offset term). In (14), the sample error is for the function $\pi(f_{\mathbf{z}})$ instead of $f_{\mathbf{z}}$ while the misclassification error (5) is kept: $\mathcal{R}(\mathrm{sgn}(\pi(f_{\mathbf{z}}))) = \mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}}))$. Since the bound for $V(y, \pi(f)(x))$ is much smaller than that for $V(y, f(x))$, the projection improves the sample error estimate.

Based on estimates for the regularization error and sample error above, our error analysis will provide $\varepsilon(\delta, m, C, \beta) > 0$ for any $0 < \delta < 1$ such that with confidence $1 - \delta$,

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q) \le (1+\beta)\{\mathcal{D}(C) + \varepsilon(\delta, m, C, \beta)\}. \tag{20}$$

Here $0 < \beta \le 1$ is an arbitrarily fixed number. Moreover, $\lim\limits_{m\to\infty} \varepsilon(\delta, m, C, \beta) = 0$.

If f_q lies in the $L^q_{\rho_X}$-closure of $\overline{\mathcal{H}_K}$, then $\lim\limits_{t\to 0} \mathcal{K}(f_q, t) = 0$ by (19). Hence $\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q) \to 0$ with confidence as m (and hence $C = C(m)$) becomes large. This is the case when K is a universal kernel, i.e., $\mathcal{H}_K$ is dense in $C(X)$, or when a sequence of kernels whose RKHS tends to be dense (e.g. polynomial kernels with increasing degrees) is used.

In summary, estimating the excess misclassification error $\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c)$ consists of three parts: the comparison (12), the regularization error $\mathcal{D}(C)$ and the sample error $\varepsilon(\delta, m, C, \beta)$ in (20). As functions of the variable C, $\mathcal{D}(C)$ decreases while $\varepsilon(\delta, m, C, \beta)$ usually increases. Choosing suitable values for the regularization parameter C will give the optimal convergence rate. To this end, we need to consider the tradeoff between these two errors. This can be done by minimizing the right side of (20), as shown in the following form.

Lemma 5 *Let $p,\alpha,\tau>0$. Denote $c_{p,\alpha,\tau}:=(p/\tau)^{\frac{\tau}{\tau+p}}+(\tau/p)^{\frac{p}{\tau+p}}$. Then for any $C>0$,*

$$C^{-p}+\frac{C^{\tau}}{m^{\alpha}}\geq c_{p,\alpha,\tau}\left(\frac{1}{m}\right)^{\frac{\alpha p}{\tau+p}}. \tag{21}$$

The equality holds if and only if $C=(p/\tau)^{\frac{1}{\tau+p}}m^{\frac{\alpha}{\tau+p}}$. This yields the optimal power $\frac{\alpha p}{\tau+p}$.

The goal of the regularization error estimates is to have p in $\mathcal{D}(C)=O(C^{-p})$, as large as possible. But $p\leq 1$ according to Proposition 4. Good methods for sample error estimates provide large α and small τ such that $\varepsilon(\delta,m,C,\beta)=O(\frac{C^{\tau}}{m^{\alpha}})$. Notice that, as always, both the approximation properties (represented by the exponent p) and the estimation properties (represented by the exponents τ and α) are important to get good estimates for the learning rates (with the optimal rate $\frac{\alpha p}{\tau+p}$).

2. Demonstrating with Weakly Separable Distributions

With some special cases let us demonstrate how our error analysis yields some guidelines for choosing the regularization parameter C. For weakly separable distributions, we also compare our results with bounds in the literature. To this end, we need the covering number of the unit ball $\mathcal{B}:=\{f\in\mathcal{H}_K:\|f\|_K\leq 1\}$ of $\mathcal{H}_K$ (considered as a subset of $C(X)$).

Definition 6 *For a subset $\mathcal{F}$ of a metric space and $\eta>0$, the* covering number $\mathcal{N}(\mathcal{F},\eta)$ *is defined to be the minimal integer $\ell\in\mathbb{N}$ such that there exist ℓ disks with radius η covering $\mathcal{F}$.*

Denote the covering number of $\mathcal{B}$ in $C(X)$ by $\mathcal{N}(\mathcal{B},\eta)$. Recall the constant $\widetilde{\kappa}$ defined in (18). Since the algorithm involves an offset term, we also need its covering number and set

$$\mathcal{N}(\eta):=\left\{(\kappa+\frac{1}{\widetilde{\kappa}})\frac{1}{\eta}+1\right\}\mathcal{N}(\mathcal{B},\eta). \tag{22}$$

Definition 7 *We say that the Mercer kernel K has* logarithmic complexity exponent $s\geq 1$ *if for some $c>0$, there holds*

$$\log\mathcal{N}(\eta)\leq c\,(\log(1/\eta))^{s},\qquad\forall\eta>0. \tag{23}$$

The kernel K has polynomial complexity exponent $s>0$ *if*

$$\log\mathcal{N}(\eta)\leq c\,(1/\eta)^{s},\qquad\forall\eta>0. \tag{24}$$

The covering number $\mathcal{N}(\mathcal{B},\eta)$ has been extensively studied (see e.g. Bartlett, 1998; Williamson, Smola and Schölkopf, 2001; Zhou, 2002; Zhou, 2003a). In particular, for convolution type kernels $K(x,y)=k(x-y)$ with $\hat{k}\geq 0$ decaying exponentially fast, (23) holds (Zhou, 2002, Theorem 3). As an example, consider the Gaussian kernel $K(x,y)=\exp\{-|x-y|^2/\sigma^2\}$ with $\sigma>0$. If $X\subset[0,1]^n$ and $0<\eta\leq\exp\{90n^2/\sigma^2-11n-3\}$, then (23) is valid (Zhou, 2002) with $s=n+1$. A lower bound (Zhou, 2003a) holds with $s=\frac{n}{2}$, which shows the upper bound is almost sharp. It was also shown in (Zhou, 2003a) that K has polynomial complexity exponent $2n/p$ if K is C^p.

To demonstrate our main results concerning the regularization parameter C, we shall only choose kernels having logarithmic complexity exponents or having polynomial complexity exponents with s small. This means, $\overline{\mathcal{H}_K}$ has small complexity.

The special case we consider here is the deterministic case: $\mathcal{R}(f_c) = 0$. Understanding how to find $\mathcal{D}(C)$ and $\varepsilon(\delta, m, C, \beta)$ in (20) and then how to choose the parameter C is our target. For $M \geq 0$, denote

$$\theta_M = \begin{cases} 0, & \text{if } M = 0, \\ 1, & \text{if } M > 0. \end{cases} \tag{25}$$

Proposition 8 *Suppose $\mathcal{R}(f_c) = 0$. If $f_{K,C} \in \overline{\mathcal{H}_K}$ satisfies $\|V(y, f_{K,C}(x))\|_\infty \leq M$, then for every $0 < \delta < 1$, with confidence at least $1 - \delta$, we have*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) \leq 2 \max\left\{\varepsilon^*, \frac{4M \log(2/\delta)}{m}\right\} + 4\mathcal{D}(C), \tag{26}$$

where with $\mathcal{M} := \sqrt{2C\mathcal{D}(C) + 2C\varepsilon\theta_M}$, $\varepsilon^* > 0$ is the unique solution to the equation

$$\log \mathcal{N}\left(\frac{\varepsilon}{q 2^{q+3} \mathcal{M}}\right) - \frac{3m\varepsilon}{2^{q+9}} = \log(\delta/2). \tag{27}$$

(a) *If (23) is valid, then with $\widetilde{c} = 2^{q+9}\{1 + c((q+4)\log 8)^s\}$,*

$$\varepsilon^* \leq \widetilde{c}\left(\frac{(\log m + \log(C\mathcal{D}(C)) + \log(C\theta_M))^s + \log(2/\delta)}{m}\right).$$

(b) *If (24) is valid and $C \geq 1$, then with a constant $\widetilde{c}$ (given explicitly in the proof),*

$$\varepsilon^* \leq \widetilde{c} \log(2/\delta)\left\{\frac{(2C\mathcal{D}(C))^{s/(2s+2)}}{m^{1/(s+1)}} + \frac{(2C)^{s/(s+2)}}{m^{1/(\frac{s}{2}+1)}}\right\}. \tag{28}$$

Note that the above constant $\widetilde{c}$ depends on c, q, and κ, but not on C, m or δ. The proof of Proposition 8 will be given in Section 5.

We can now derive our error bound for weakly separable distributions.

Definition 9 *We say that ρ is* (weakly) separable *by $\overline{\mathcal{H}_K}$ if there is a function $f_{sp} \in \overline{\mathcal{H}_K}$, called a* separating function, *satisfying $\|f_{sp}^*\|_K = 1$ and $y f_{sp}(x) > 0$ almost everywhere. It has* separation exponent $\theta \in (0, +\infty]$ *if there are positive constants γ, c' such that*

$$\rho_X\{x \in X : |f_{sp}(x)| < \gamma t\} \leq c' t^\theta, \qquad \forall\, t > 0. \tag{29}$$

Observe that condition (29) with $\theta = +\infty$ is equivalent to

$$\rho_X\{x \in X : |f_{\mathrm{sp}}(x)| < \gamma t\} = 0, \qquad \forall\, 0 < t < 1.$$

That is, $|f_{\mathrm{sp}}(x)| \geq \gamma$ almost everywhere. Thus, weakly separable distributions with separation exponent $\theta = +\infty$ are exactly strictly separable distributions. Recall (e.g. Vapnik, 1998; Shawe-Taylor et al., 1998) that ρ is said to be *strictly separable* by $\overline{\mathcal{H}_K}$ with margin $\gamma > 0$ if ρ is (weakly) separable together with the requirement $y f_{\mathrm{sp}}(x) \geq \gamma$ almost everywhere.

¿From the first part of Definition 9, we know that for a separable distribution ρ, the set $\{x : f_{\mathrm{sp}}(x) = 0\}$ has ρ_X-measure zero, hence

$$\lim_{t \to 0} \rho_X\{x \in X : |f_{\mathrm{sp}}(x)| < t\} = 0.$$

The separation exponent θ measures the asymptotic behavior of ρ near the boundary of two classes. It gives the convergence with a polynomial decay.

Theorem 10 *If ρ is separable and has separation exponent $\theta \in (0,+\infty]$ with (29) valid, then for every $0<\delta<1$, with confidence at least $1-\delta$, we have*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) \le 2\varepsilon^* + \frac{8\log(2/\delta)}{m} + 4(2c')^{\frac{2}{\theta+2}}(C\gamma^2)^{-\frac{\theta}{\theta+2}}, \tag{30}$$

where ε^* is solved by

$$\log \mathcal{N}\left(\frac{\varepsilon}{q2^{q+3}\sqrt{2(2c')^{\frac{2}{\theta+2}}\gamma^{-\frac{2\theta}{\theta+2}}C^{\frac{2}{\theta+2}}+2C\varepsilon}}\right) - \frac{3m\varepsilon}{2^{q+9}} = \log(\delta/2). \tag{31}$$

(a) *If (23) is satisfied, with a constant $\widetilde{c}$ depending on γ we have*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) \le \widetilde{c}\left(\frac{\log(2/\delta)+(\log m+\log C)^s}{m} + C^{-\frac{\theta}{\theta+2}}\right). \tag{32}$$

(b) *If (24) is satisfied, there holds*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) \le \widetilde{c}\left\{\log\frac{2}{\delta}\left(C^{\frac{s}{(s+1)(\theta+2)}}m^{-\frac{1}{s+1}} + C^{\frac{s}{s+2}}m^{-\frac{1}{s/2+1}}\right) + C^{-\frac{\theta}{\theta+2}}\right\}. \tag{33}$$

Proof Choose $t = (\frac{\gamma^\theta}{2c'C})^{1/(\theta+2)} > 0$ and the function $f_{K,C} = \frac{1}{t}f_{sp} \in \overline{\mathcal{H}}_K$. Then we have $\frac{1}{2C}\|f^*_{K,C}\|^2_K = \frac{1}{2Ct^2}$.

Since $yf_{sp}(x) > 0$ almost everywhere, we know that for almost every $x \in X$, $y = \mathrm{sgn}(f_{sp})(x)$. This means $f_\rho = \mathrm{sgn}(f_{sp})$ and then $f_q = \mathrm{sgn}(f_{sp})$. It follows that $\mathcal{R}(f_c) = 0$ and $V(y, f_{K,C}(x)) = (1-\frac{|f_{sp}(x)|}{t})^q_+ \in [0,1]$ almost everywhere. Therefore, we may take $M=1$ in Proposition 8. Moreover,

$$\mathcal{E}(f_{K,C}) = \int_Z \Big(1-\frac{yf_{sp}(x)}{t}\Big)^q_+ d\rho = \int_X \Big(1-\frac{|f_{sp}(x)|}{t}\Big)^q_+ d\rho_X.$$

This can be bounded by (29) as

$$\int_{\{x:|f_{sp}(x)|<t\}} \Big(1-\frac{|f_{sp}(x)|}{t}\Big)^q_+ d\rho_X \le \rho_X\{x\in X : |f_{sp}(x)|<t\} \le c'\Big(\frac{t}{\gamma}\Big)^\theta.$$

It follows from the choice of t that

$$\mathcal{D}(C) \le c'\Big(\frac{t}{\gamma}\Big)^\theta + \frac{1}{2Ct^2} = (2c')^{\frac{2}{\theta+2}}(C\gamma^2)^{-\frac{\theta}{\theta+2}}. \tag{34}$$

Then our conclusion follows from Proposition 8. ■

In case (a), the bound (32) tells us to choose C such that $(\log C)^s/m \to 0$ and $C\to\infty$ as $m\to\infty$. By (32) a reasonable choice of the regularization parameter C is $C = m^{\frac{\theta+2}{\theta}}$, which yields $\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) = O(\frac{(\log m)^s}{m})$. In case (b),

$$\text{when (24) is valid,}\quad C = m^{\frac{\theta+2}{\theta+s+\theta s}} \Longrightarrow \mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) = O(m^{-\frac{\theta}{\theta+s+\theta s}}). \tag{35}$$

The following is a typical example of separable distribution which is not strictly separable. In this example, the separation exponent is $\theta = 1$.

Example 1 *Let $X = [-1/2, 1/2]$ and ρ be the Borel probability measure on Z such that ρ_X is the Lebesgue measure on X and*

$$f_\rho(x) = \begin{cases} 1, & \text{for } -1/2 \le x \le -1/4 \text{ and } 1/4 \le x \le 1/2, \\ -1, & \text{for } -1/4 \le x < 1/4. \end{cases}$$

(a) *If K is the linear polynomial kernel $K(x,y) = x \cdot y$, then $\mathcal{R}(sgn(f_{\mathbf{z}})) - \mathcal{R}(f_c) \ge 1/4$.*

(b) *If K is the quadratic polynomial kernel $K(x,y) = (x \cdot y)^2$, then with confidence $1-\delta$,*

$$\mathcal{R}(sgn(f_{\mathbf{z}})) \le \frac{q(q+4)2^{q+11}\,(\log m + \log C + 2\log(2/\delta))}{m} + \frac{16}{C^{1/3}}.$$

Hence one should take C such that $\log C/m \to 0$ and $C \to \infty$ as $m \to \infty$.

Proof The first statement is trivial.

To see (b), we note that $\dim \mathcal{H}_K = 1$ and $\kappa = 1/4$. Also, $\mathcal{E}_0 = \inf_{b \in \mathbb{R}} \mathcal{E}(b) = 1$. Hence $\widetilde{\kappa} = 1/(1+q4^{q-2})$. Since $\mathcal{H}_K = \{ax^2 : a \in \mathbb{R}\}$ and $\|ax^2\|_K = |a|$, $\mathcal{N}(\mathcal{B}, \eta) \le 1/(2\eta)$. Then (23) holds with $s = 1$ and $c = 4q$. By Proposition 8, we see that $\varepsilon^* \le \frac{q(q+4)2^{q+11}(\log(Cm)+\log(2/\delta))}{m}$.

Take the function $f_{\rm sp}(x) = x^2 - 1/16 \in \overline{\mathcal{H}_K}$ with $\|f_{\rm sp}^*\|_K = 1$. We see that $y f_{\rm sp}(x) > 0$ almost everywhere. Moreover,

$$\{x \in X : |f_{\rm sp}(x)| < t\} \subseteq \left[\frac{\sqrt{1-16t}}{4}, \frac{\sqrt{1+16t}}{4}\right] \bigcup \left[-\frac{\sqrt{1+16t}}{4}, -\frac{\sqrt{1-16t}}{4}\right].$$

The measure of this set is bounded by $8\sqrt{2}t$. Hence (29) holds with $\theta = 1$, $\gamma = 1$ and $c' = 8\sqrt{2}$. Then Theorem 10 gives the stated bound for $\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}}))$. ■

In this paper, for the sample error estimates we only use the (uniform) covering numbers in $C(X)$. Within the last a few years, the empirical covering numbers (in ℓ^∞ or ℓ^2), the leave-one-out error or stability analysis (e.g. Vapnik, 1998; Bousquet and Ellisseeff, 2002; Zhang, 2004), and some other advanced empirical process techniques such as the local Rademacher averages (van der Vaart and Wellner, 1996; Bartlett, Bousquet and Mendelson, 2004; and references therein) and the entropy integrals (van der Vaart and Wellner, 1996) have been developed to get better sample error estimates. These techniques can be applied to various learning algorithms. They are powerful to handle general hypothesis spaces even with large capacity.

In Zhang (2004) the leave-one-out technique was applied to improve the sample error estimates given in Bousquet and Ellisseeff (2002): the sample error has a kernel-independent bound $O(\frac{C}{m})$, improving the bound $O(\frac{C}{\sqrt{m}})$ in Bousquet and Ellisseeff (2002); while the regularization error $\widetilde{\mathcal{D}}(C)$ depends on K and ρ. In particular, for $q = 2$ the bound (Zhang, 2004, Corollary 4.2) takes the form:

$$E(\mathcal{E}(f_{\mathbf{z}})) \le \left(1 + \frac{4\kappa^2 C}{m}\right)^2 \inf_{f \in \mathcal{H}_K} \left\{\mathcal{E}(f) + \frac{1}{2C}\|f\|_K^2\right\} = \left(1 + \frac{4\kappa^2 C}{m}\right)^2 \left\{\widetilde{\mathcal{D}}(C) + \mathcal{E}(f_q)\right\}. \quad (36)$$

In Bartlett, Jordan and McAuliffe (2003) the empirical process techniques are used to improve the sample error estimates for ERM. In particular, Theorem 12 there states that for a convex set $\mathcal{F}$ of functions on X, the minimizer $\widehat{f}$ of the empirical error over $\mathcal{F}$ satisfies

$$\mathcal{E}(\widehat{f}) \le \inf_{f \in \mathcal{F}} \mathcal{E}(f) + K \max\left\{\varepsilon^*, \left(\frac{c_r L^2 \log(1/\delta)}{m}\right)^{1/(2-\beta)}, \frac{BL\log(1/\delta)}{m}\right\}. \quad (37)$$

Here K, c_r, β are constants, and ε^* is solved by an inequality. The constant B is a bound for differences, and L is the Lipschitz constant for the loss ϕ with respect to a pseudometric on $\mathbb{R}$. For more details, see Bartlett, Jordan and McAuliffe (2003). The definitions of B and L together tell us that

$$|\phi(y_1 f(x_1)) - \phi(y_2 f(x_2))| \le LB, \qquad \forall\, (x_1, y_1) \in Z, (x_2, y_2) \in Z, f \in \mathcal{F}. \tag{38}$$

Because of our improvement for the bound of the random variables $V(y, f(x))$ given by the projection operator, for the algorithm (3) the error bounds we derive here are better than existing results when the kernel has small complexity. Let us confirm this for separable distributions with separation exponent $0 < \theta < \infty$ and for kernels with polynomial complexity exponent $s > 0$ satisfying (24). Recall that $\mathcal{D}(C) = O(C^{-\frac{\theta}{\theta+2}})$. Take $q = 2$.

Consider the estimate (36). This together with (34) tells us that the derived bound for $E(\mathcal{E}(f_{\mathbf{z}}) - \mathcal{E}(f_q))$ is at least of the order $\widetilde{\mathcal{D}}(C) + \frac{C}{m}\widetilde{\mathcal{D}}(C) = O(C^{-\frac{\theta}{\theta+2}} + \frac{C^{\frac{2}{\theta+2}}}{m})$. By Lemma 5, the optimal bound derived from (36) is $O(m^{-\frac{\theta}{\theta+2}})$. Thus, our bound (35) is better than (36) when $0 < s < \frac{2}{\theta+1}$.

Turn to the estimate (37). Take $\mathcal{F}$ to be the ball of radius $\sqrt{2C\widetilde{\mathcal{D}}(C)}$ of $\mathcal{H}_K$ (the expected smallest ball where $f_{\mathbf{z}}^*$ lies), we find that the constant LB in (38) should be at least $\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)}$. This tells us that one term in the bound (37) for the sample error is at least $O\Big(\frac{\sqrt{2C\widetilde{\mathcal{D}}(C)}}{m}\Big) = O(\frac{C^{\frac{1}{\theta+2}}}{m})$, while the other two terms are more involved. Applying Lemma 5 again, we find that the bound derived from (37) is at least $O(m^{-\frac{\theta}{\theta+1}})$. So our bound (35) is better than (37) at least for $0 < s < \frac{1}{\theta+1}$.

Thus, our analysis with the help of the projection operator improves existing error bounds when the kernel has small complexity. Note that in (35), the values of s for which the projection operator gives an improvement are values corresponding to rapidly diminishing regularization ($C = m^\beta$ with $\beta > 1$ being large).

For kernels with large complexity, refined empirical process techniques (e.g. van der Vaart and Wellner, 1996; Mendelson, 2002; Zhang, 2004; Bartlett, Jordan and McAuliffe, 2003) should give better bounds: the capacity of the hypothesis space may have more influence on the sample error than the bound of the random variable $V(y, f(x))$, and the entropy integral is powerful to reduce this influence. To find the range of this large complexity, one needs explicit rate analysis for both the sample error and regularization error. This is out of our scope. It would be interesting to get better error bounds by combining the ideas of empirical process techniques and the projection operator.

Problem 11 *How much improvement can we get for the total error when we apply the projection operator to empirical process techniques?*

Problem 12 *Given $\theta > 0, s > 0, q \ge 1$, and $0 < \delta < 1$, what is the largest number $\alpha > 0$ such that $\mathcal{R}(sgn(f_{\mathbf{z}})) = O(m^{-\alpha})$ with confidence $1 - \delta$ whenever the distribution ρ is separable with separation exponent θ and the kernel K satisfies (24)? What is the corresponding optimal choice of the regularization parameter C?*

In particular, for Example 1 we have the following.

Conjecture 13 *In Example 1, with confidence $1 - \delta$ there holds $\mathcal{R}(sgn(f_{\mathbf{z}})) = O(\frac{\log(2/\delta)}{m})$ by choosing $C = m^3$.*

Consider the well known setting (Vapnik, 1998; Shawe-Taylor et al., 1998; Steinwart, 2001; Cristianini and Shawe-Taylor, 2000) of strictly separable distributions with margin $\gamma > 0$. In this case, Shawe-Taylor et al. (1998) shows that

$$\mathcal{R}(\mathrm{sgn}(f)) \le \frac{2}{m}(\log \mathcal{N}(\mathcal{F}, 2m, \gamma/2) + \log(2/\delta)) \tag{39}$$

with confidence $1-\delta$ for $m > 2/\gamma$ whenever $f \in \mathcal{F}$ satisfies $y_i f(x_i) \ge \gamma$. Here $\mathcal{F}$ is a function set, $\mathcal{N}(\mathcal{F}, 2m, \gamma/2) = \sup_{\vec{t} \in X^{2m}} \mathcal{N}(\mathcal{F}, \vec{t}, \gamma/2)$ and $\mathcal{N}(\mathcal{F}, \vec{t}, \gamma/2)$ denotes the covering number of the set $\{(f(t_i))_{i=1}^{2m} : f \in \mathcal{F}\}$ in $\mathbb{R}^{2m}$ with the ℓ^∞ metric. For a comparison of this covering number with that in $C(X)$, see Pontil (2003).

In our analysis, we can take $f_{K,C} = \frac{1}{\gamma} f_{\mathrm{sp}} \in \overline{\mathcal{H}_K}$. Then $M = \|V(y, f_{K,C}(x))\|_\infty = 0$, and $\mathcal{D}(C) = \frac{1}{2C\gamma^2}$. We see from Proposition 8 (a) that when (23) is satisfied, there holds $\mathcal{R}(f_{\mathbf{z}}) \le \widetilde{c}(\frac{(\log(1/\gamma)+\log m)^s + \log(1/\delta)}{m})$. But the optimal bound $O(\frac{1}{m})$ is also valid for spaces with larger complexity, which can be seen from (39) by the relation of the covering numbers: $\mathcal{N}(\mathcal{F}, 2m, \gamma/2) \le \mathcal{N}(\mathcal{F}, \gamma/2)$. See also Steinwart (2001). We see the shortcoming of our approach for kernels with large complexity. This convinces the interest of Problem 11 raised above.

3. Comparison of Errors

In this section we consider how to bound the misclassification error by the V-risk. A systematic study of this problem was done for convex loss functions by Zhang (2004), and for more general loss functions by Bartlett, Jordan, and McAuliffe (2003). Using these results, it can be shown that for the loss function V_q, there holds

$$\psi\Big(\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c)\Big) \le \mathcal{E}(f) - \mathcal{E}(f_q),$$

where $\psi : [0,1] \to \mathbb{R}_+$ is a function defined in Bartlett, Jordan, and McAuliffe (2003) and can be explicitly computed.

In fact, we have

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) \le c\sqrt{\mathcal{E}(f) - \mathcal{E}(f_q)}. \tag{40}$$

Such a comparison of errors holds true even for a general convex loss function, see Theorem 34 in Appendix. The derived bound for the constant c in (40) need not be optimal. In the following we shall give an optimal estimate in a simpler form.

The constant we derive depends on q and is given by

$$C_q = \begin{cases} 1, & \text{if } 1 \le q \le 2, \\ 2(q-1)/q, & \text{if } q > 2. \end{cases} \tag{41}$$

We can see that $C_q \le 2$.

Theorem 14 *Let $f : X \to \mathbb{R}$ be measurable. Then*

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) \le \sqrt{C_q\,(\mathcal{E}(f) - \mathcal{E}(f_q))} \le \sqrt{2\,(\mathcal{E}(f) - \mathcal{E}(f_q))}.$$

Proof By the definition, only points with $\mathrm{sgn}(f)(x) \neq f_c(x)$ are involved for the misclassification error. Hence

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) = \int_X |f_\rho(x)| \chi_{\{\mathrm{sgn}(f)(x) \neq \mathrm{sgn}(f_q)(x)\}} d\rho_X. \tag{42}$$

This in connection with the Schwartz inequality implies that

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) \leq \left\{ \int_X |f_\rho(x)|^2 \chi_{\{\mathrm{sgn}(f)(x) \neq \mathrm{sgn}(f_q)(x)\}} d\rho_X \right\}^{1/2}.$$

Thus it is sufficient to show for those $x \in X$ with $\mathrm{sgn}(f)(x) \neq \mathrm{sgn}(f_q)(x)$,

$$|f_\rho(x)|^2 \leq C_q \left(\mathcal{E}(f|x) - \mathcal{E}(f_q|x) \right), \tag{43}$$

where for $x \in X$, we have denoted

$$\mathcal{E}(f|x) := \int_Y V_q(y, f(x)) d\rho(y|x). \tag{44}$$

By the definition of the loss function V_q, we have

$$\mathcal{E}(f|x) = (1 - f(x))_+^q \frac{1 + f_\rho(x)}{2} + (1 + f(x))_+^q \frac{1 - f_\rho(x)}{2}. \tag{45}$$

It follows that

$$\mathcal{E}(f|x) - \mathcal{E}(f_q|x) = \int_0^{f(x) - f_q(x)} F(u) du, \tag{46}$$

where $F(u)$ is the function (depending on the parameter $f_\rho(x)$):

$$F(u) = \frac{1 - f_\rho(x)}{2} q (1 + f_q(x) + u)_+^{q-1} - \frac{1 + f_\rho(x)}{2} q (1 - f_q(x) - u)_+^{q-1}, \quad u \in \mathbb{R}.$$

Since $F(u)$ is nondecreasing and $F(0) = 0$, we see from (46) that when $\mathrm{sgn}(f)(x) \neq \mathrm{sgn}(f_q)(x)$, there holds

$$\mathcal{E}(f|x) - \mathcal{E}(f_q|x) \geq \int_0^{-f_q(x)} F(u) du = \mathcal{E}(0|x) - \mathcal{E}(f_q|x) = 1 - \mathcal{E}(f_q|x). \tag{47}$$

But

$$\mathcal{E}(f_q|x) = \frac{2^{q-1} \left(1 - |f_\rho(x)|^2\right)}{\left\{ (1 + f_\rho(x))^{1/(q-1)} + (1 - f_\rho(x))^{1/(q-1)} \right\}^{q-1}}. \tag{48}$$

Therefore, (43) is valid (with $t = f_\rho(x)$) once the following inequality is verified:

$$\frac{t^2}{C_q} \leq 1 - \frac{2^{q-1} \left(1 - t^2\right)}{\left\{ (1+t)^{1/(q-1)} + (1-t)^{1/(q-1)} \right\}^{q-1}}, \qquad t \in [-1, 1].$$

This is the same as the inequality

$$\left(1 - \frac{t^2}{C_q}\right)^{-1/(q-1)} \leq \frac{(1+t)^{-1/(q-1)} + (1-t)^{-1/(q-1)}}{2}, \qquad t \in [-1, 1]. \tag{49}$$

To prove (49), we use the Taylor expansion

$$(1+u)^{\alpha} = \sum_{k=0}^{\infty} \binom{\alpha}{k} u^k = \sum_{k=0}^{\infty} \frac{\prod_{\ell=0}^{k-1}(\alpha-\ell)}{k!}\, u^k, \qquad u \in (-1,1).$$

With $\alpha = -1/(q-1)$, we have

$$\left(1-\frac{t^2}{C_q}\right)^{-1/(q-1)} = \sum_{k=0}^{\infty} \frac{\prod_{\ell=0}^{k-1}\left(\frac{1}{q-1}+\ell\right)}{k!}\, \frac{1}{C_q^k}\, t^{2k}$$

and (all the odd power terms vanish)

$$\frac{(1+t)^{-1/(q-1)}+(1-t)^{-1/(q-1)}}{2} = \sum_{k=0}^{\infty} \frac{\prod_{\ell=0}^{k-1}\left(\frac{1}{q-1}+\ell\right)}{k!} \left(\prod_{\ell=0}^{k-1} \frac{\frac{1}{q-1}+\ell+k}{1+\ell+k}\right) t^{2k}.$$

Note that

$$\prod_{\ell=0}^{k-1} \frac{\frac{1}{q-1}+\ell+k}{1+\ell+k} \geq \frac{1}{C_q^k}.$$

This proves (49) and hence our conclusion. ■

When $\mathcal{R}(f_c) = 0$, the estimate in Theorem 14 can be improved (Zhang 2004, Bartlett, Jordan and McAuliffe 2003) to

$$\mathcal{R}(\operatorname{sgn}(f)) \leq \mathcal{E}(f). \tag{50}$$

In fact, $\mathcal{R}(f_c) = 0$ implies $|f_\rho(x)| = 1$ almost everywhere. This in connection with (48) and (47) gives $\mathcal{E}(f_q|x) = 0$ and $\mathcal{E}(f|x) \geq 1$ when $\operatorname{sgn}(f)(x) \neq f_c(x)$. Then (43) can be improved to the form $|f_\rho(x)| \leq \mathcal{E}(f|x)$. Hence (50) follows from (42).

Theorem 14 tells us that the misclassification error can be bounded by the V-risk associated with the loss function V. So our next step is to study the convergence of the q-classifier with respect to the V-risk $\mathcal{E}$.

4. Bounding the Offset

If the offset b is fixed in the scheme (3), the sample error can be bounded by standard argument using some measurements of the capacity of the RKHS. However, the offset is part of the optimization problem and even its boundedness can not be seen from the definition. This makes our setting here essentially different from the standard Tikhonov regularization scheme. The difficulty of bounding the offset b has been realized in the literature (e.g. Steinwart, 2002; Bousquet and Ellisseeff, 2002). In this paper we shall overcome this difficulty by means of special features of the loss function V.

The difficulty raised by the offset can also be seen from the stability analysis (Bousquet and Ellisseeff, 2002). As shown in Bousquet and Ellisseeff (2002), the SVM 1-norm classifier without the offset is uniformly stable, meaning that $\sup_{\mathbf{z}\in Z^m, z_0'\in Z} \|V(y, f_{\mathbf{z}}(x)) - V(y, f_{\mathbf{z}'}(x))\|_{L^\infty} \leq \beta_m$ with $\beta_m \to 0$ as $m \to \infty$, and $\mathbf{z}'$ is the same as $\mathbf{z}$ except that one element of $\mathbf{z}$ is replaced by z_0'.

The SVM 1-norm classifier with the offset is not uniformly stable. To see this, we choose $x_0 \in X$ and samples $\mathbf{z} = \{(x_0, y_i)\}_{i=1}^{2n+1}$ with $y_i = 1$ for $i = 1, \ldots, n+1$, and $y_i = -1$ for $i = n+2, \ldots, 2n+1$.

Take $z_0' = (x_0, -1)$. As x_i are identical, one can see from the definition (9) that $f_{\mathbf{z}}^* = 0$ (since $\mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}}) = \mathcal{E}_{\mathbf{z}}(b_{\mathbf{z}} + f_{\mathbf{z}}(x_0))$). It follows that $f_{\mathbf{z}} = 1$ while $f_{\mathbf{z}'} = -1$. Thus, $|f_{\mathbf{z}} - f_{\mathbf{z}'}| = 2$ which does not converge to zero as $m = 2n+1$ tends to infinity. It is unknown whether the q-norm classifier is uniformly stable for $q > 1$.

How to bound the offset is the main goal of this section. In Wu and Zhou (2004) a direct computation is used to realize this point for $q = 1$. Here the index $q > 1$ makes a direct computation very difficult, and we shall use two bounds to overcome this difficulty. By $x \in (X, \rho_X)$ we mean that x lies in the support of the measure ρ_X on X.

Lemma 15 *For any $C > 0, m \in \mathbb{N}$ and $\mathbf{z} \in Z^m$, a minimizer of (9) satisfies*

$$\min_{1\le i\le m} f_{\mathbf{z}}(x_i) \le 1 \quad \textit{and} \quad \max_{1\le i\le m} f_{\mathbf{z}}(x_i) \ge -1 \tag{51}$$

and a minimizer of (15) satisfies

$$\inf_{x\in(X,\rho_X)} \widetilde{f}_{K,C}(x) \le 1 \quad \textit{and} \quad \sup_{x\in(X,\rho_X)} \widetilde{f}_{K,C}(x) \ge -1. \tag{52}$$

Proof Suppose a minimizer of (9) $f_{\mathbf{z}}$ satisfies $r := \min_{1\le i\le m} f_{\mathbf{z}}(x_i) > 1$. Then $f_{\mathbf{z}}(x_i) - (r-1) \ge 1$ for each i. We claim that

$$y_i = 1, \qquad \forall i = 1, \dots, m.$$

In fact, if the set $I := \{i \in \{1, \dots, m\} : y_i = -1\}$ is not empty, we have

$$\mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}} - (r-1)) = \frac{1}{m}\sum_{i\in I}(1 + f_{\mathbf{z}}(x_i) - (r-1))^q < \frac{1}{m}\sum_{i\in I}(1 + f_{\mathbf{z}}(x_i))^q = \mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}}),$$

which is a contradiction to the definition of $f_{\mathbf{z}}$. Hence our claim is verified. From the claim we see that $\mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}} - (r-1)) = 0 = \mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}})$. This tells us that $\widetilde{f}_{\mathbf{z}} := f_{\mathbf{z}} - (r-1)$ is a minimizer of (9) satisfying the first inequality, hence both inequalities of (51).

In the same way, if a minimizer of (9) $f_{\mathbf{z}}$ satisfies $r := \max_{1\le i\le m} f_{\mathbf{z}}(x_i) < -1$. Then we can see that $y_i = -1$ for each i. Hence $\mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}} - r - 1) = \mathcal{E}_{\mathbf{z}}(f_{\mathbf{z}})$ and $\widetilde{f}_{\mathbf{z}} := f_{\mathbf{z}} - r - 1$ is a minimizer of (9) satisfying the second inequality and hence both inequalities of (51).

Therefore, we can always find a minimizer of (9) satisfying (51).

We prove the second statement in the same way. Suppose $r := \inf_{x\in(X,\rho_X)} \widetilde{f}_{K,C}(x) > 1$ for a minimizer $\widetilde{f}_{K,C}$ of (15). Then $\widetilde{f}_{K,C}(x) - (r-1) \ge 1$ for almost every $x \in (X, \rho_X)$. Hence

$$\mathcal{E}\left(\widetilde{f}_{K,C} - (r-1)\right) = \int_X \left(1 + \widetilde{f}_{K,C}(x) - (r-1)\right)^q P(\mathcal{Y} = -1|x) d\rho_X \le \mathcal{E}(\widetilde{f}_{K,C}).$$

As $f_{K,C}$ is a minimizer of (15), the above equality must hold. It follows that $P(\mathcal{Y} = -1|x) = 0$ for almost every $x \in (X, \rho_X)$. Hence $\widetilde{F}_{K,C} := \widetilde{f}_{K,C} - (r-1)$ is a minimizer of (15) satisfying the first inequality and thereby both inequalities of (52).

Similarly, when $r := \sup_{x\in(X,\rho_X)} \widetilde{f}_{K,C}(x) < -1$ for a minimizer $\widetilde{f}_{K,C}$ of (15). Then $P(\mathcal{Y} = 1|x) = 0$ for almost every $x \in (X, \rho_X)$. Hence $\widetilde{F}_{K,C} := f_{K,C} - r - 1$ is a minimizer of (15) satisfying the second inequality and thereby both inequalities of (52).

Thus, (52) can always be realized by a minimizer of (15). ■

In what follows we always choose $f_{\mathbf{z}}$ and $\widetilde{f}_{K,C}$ to satisfy (51) and (52), respectively.
Lemma 15 yields bounds for the $\mathcal{H}_K$-norm and offset for $f_{\mathbf{z}}$ and $\widetilde{f}_{K,C}$. Denote $b_{\widetilde{f}_{K,C}}$ as $\widetilde{b}_{K,C}$.

Lemma 16 *For any $C>0, m\in\mathbb{N}$, $f_{K,C}\in\overline{\mathcal{H}_K}$, and $\mathbf{z}\in Z^m$, there hold*

(a) $\|\widetilde{f}^*_{K,C}\|_K \le \sqrt{2C\widetilde{\mathcal{D}}(C)} \le \sqrt{2C}, \quad |\widetilde{b}_{K,C}| \le 1+\|\widetilde{f}^*_{K,C}\|_\infty.$

(b) $\|\widetilde{f}_{K,C}\|_\infty \le 1+2\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)} \le 1+2\kappa\sqrt{2C}, \quad \mathcal{E}(\widetilde{f}_{K,C}) \le \mathcal{E}(f_q)+\widetilde{\mathcal{D}}(C) \le 1.$

(c) $|b_{\mathbf{z}}| \le 1+\kappa\|f^*_{\mathbf{z}}\|_K.$

Proof By the definition (15), we see from the choice $f=0+0$ that

$$\widetilde{\mathcal{D}}(C) = \mathcal{E}(\widetilde{f}_{K,C}) - \mathcal{E}(f_q) + \frac{1}{2C}\|\widetilde{f}^*_{K,C}\|^2_K \le 1-\mathcal{E}(f_q). \tag{53}$$

Then the first inequality in (a) follows.

Note that (52) gives

$$-1 \le \sup_{x\in(X,\rho_X)} \widetilde{f}_{K,C} \le \widetilde{b}_{K,C} + \|\widetilde{f}^*_{K,C}\|_\infty$$

and

$$\widetilde{b}_{K,C} - \|\widetilde{f}^*_{K,C}\|_\infty \le \inf_{x\in(X,\rho_X)} \widetilde{f}_{K,C} \le 1.$$

Thus, $|\widetilde{b}_{K,C}| \le 1+\|\widetilde{f}^*_{K,C}\|_\infty$. This proves the second inequality in (a).

Since the first inequality in (a) and (2) lead to $\|\widetilde{f}^*_{K,C}\|_\infty \le \kappa\sqrt{2C\widetilde{\mathcal{D}}(C)}$, we obtain $\|\widetilde{f}_{K,C}\|_\infty \le 1+2\|\widetilde{f}^*_{K,C}\|_\infty \le 1+2\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)}$. Hence the first inequality in (b) holds.

The second inequality in (b) is an easy consequence of (53).

The inequality in (c) follows from (51) in the same way as the proof of the second inequality in (a). ■

5. Convergence of the q-Norm Soft Margin Classifier

In this section, we apply the ERM technique to analyze the convergence of the q-classifier for $q>1$. The situation here is more complicated than that for $q=1$. We need the following lemma concerning $q>1$.

Lemma 17 *For $q>1$, there holds*

$$\left|(x)_+^q - (y)_+^q\right| \le q\,(\max\{x,y\})_+^{q-1}|x-y|, \qquad \forall x,y\in\mathbb{R}.$$

If $y\in[-1,1]$, then

$$\left|(1-x)_+^q - (1-y)_+^q\right| \le q4^{q-1}|x-y| + q2^{q-1}|x-y|^q, \qquad \forall x\in\mathbb{R}.$$

Proof We only need to prove the first inequality for $x > y$. This is trivial:

$$(x)_+^q - (y)_+^q = \int_y^x q(u)_+^{q-1} du \le q(x)_+^{q-1}(x-y).$$

If $y \in [-1,1]$, then $1-y \ge 0$ and the first inequality yields

$$\left|(1-x)_+^q - (1-y)_+^q\right| \le q\left(\max\{1-x,1-y\}\right)_+^{q-1}|x-y|. \tag{54}$$

When $x \ge 2y-1$, we have

$$\left|(1-x)_+^q - (1-y)_+^q\right| \le q2^{q-1}(1-y)^{q-1}|x-y| \le q4^{q-1}|x-y|.$$

When $x < 2y-1$, we have $1-x < 2(y-x)$ and $x < 1$. This in combination with $\max\{1-x,1-y\} \le \max\{1-x,2\}$ and (54) implies

$$\left|(1-x)_+^q - (1-y)_+^q\right| \le q\{(1-x)^{q-1} + 2^{q-1}\}|x-y| \le q2^{q-1}|x-y|^q + q2^{q-1}|x-y|.$$

This proves Lemma 17. ■

The second part of Lemma 17 will be used in Section 6. The first part can be used to verify Proposition 4.

Proof of Proposition 4. It is trivial that $\mathcal{D}(C) \ge \widetilde{\mathcal{D}}(C)$. To show the second inequality, apply the first inequality of Lemma 17 to the two numbers $1-y\widetilde{f}_{K,C}(x)$ and $1-y\widetilde{b}_{K,C}$. We see that

$$(1-y\widetilde{f}_{K,C}(x))_+^q \ge (1-y\widetilde{b}_{K,C})_+^q - q(1+|\widetilde{f}_{K,C}(x)|+|\widetilde{b}_{K,C}|)^{q-1}|\widetilde{f}_{K,C}^*(x)|.$$

Notice that $|\widetilde{f}_{K,C}^*(x)| \le \kappa\|\widetilde{f}_{K,C}^*\|_K$ and by Lemma 16, $|\widetilde{b}_{K,C}| \le 1+\kappa\|\widetilde{f}_{K,C}^*\|_K$. Hence

$$\mathcal{E}(\widetilde{f}_{K,C}) \ge \mathcal{E}(\widetilde{b}_{K,C}) - \kappa q 2^{q-1}(1+\kappa\|\widetilde{f}_{K,C}^*\|_K)^{q-1}\|\widetilde{f}_{K,C}^*\|_K.$$

It follows that when $\|\widetilde{f}_{K,C}^*\|_K \le \widetilde{\kappa}$, we have

$$\mathcal{E}(\widetilde{f}_{K,C}) - \mathcal{E}(f_q) \ge \mathcal{E}_0 - \kappa q 2^{q-1}(1+\kappa\widetilde{\kappa})^{q-1}\widetilde{\kappa}.$$

Since $\mathcal{E}_0 \le 1$, the definition (18) of $\widetilde{\kappa}$ yields $\widetilde{\kappa} \le 1/(1+\kappa) \le \min\{1,1/\kappa\}$. Hence

$$\widetilde{\mathcal{D}}(C) \ge \mathcal{E}(\widetilde{f}_{K,C}) - \mathcal{E}(f_q) \ge \mathcal{E}_0 - \kappa q 4^{q-1}\widetilde{\kappa} = \widetilde{\kappa} \ge \widetilde{\kappa}^2.$$

As $C \ge 1/2$, we conclude (17) in this case.

When $\|\widetilde{f}_{K,C}^*\|_K > \widetilde{\kappa}$, we also have

$$\widetilde{\mathcal{D}}(C) \ge \frac{1}{2C}\|\widetilde{f}_{K,C}^*\|_K^2 \ge \frac{\widetilde{\kappa}^2}{2C}.$$

Thus in both case we have verified (17).

Note that $\widetilde{\kappa} = 0$ if and only if $\mathcal{E}_0 = 0$. This means for some $b_0' \in [-1,1]$, $f_q(x) = b_0'$ in probability. By the definition of f_q, the last assertion of Proposition 4 follows. ■

The loss function V is not Lipschitz, but Lemma 17 enables us to derive a bound for $\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}_{\mathbf{z}}(\pi(f_{\mathbf{z}}))$ with confidence, as done for Lipschitz loss functions in Mukherjee, Rifkin and Poggio (2002). Since the function $\pi(f_{\mathbf{z}})$ changes and lies in a set of functions, we shall compare the error with the empirical error for functions from a set

$$\mathcal{F} := \{\pi(f) : f \in B_R + [-B, B]\}. \tag{55}$$

Here $B_R = \{f^* \in \mathcal{H}_K : \|f^*\|_K \le R\}$ and the constant B is a bound for the offset.

The following probability inequality was motivated by sample error estimates for the square loss (Barron 1990, Bartlett 1998, Cucker and Smale 2001, Lee, Bartlett and Williamson 1998) and will be used in our estimates.

Lemma 18 *Suppose a random variable ξ satisfies $0 \le \xi \le M$, and $\mathbf{z} = (z_i)_{i=1}^m$ are independent samples. Let $\mu = E(\xi)$. Then for every $\varepsilon > 0$ and $0 < \alpha \le 1$, there holds*

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\left\{\frac{\mu - \frac{1}{m}\sum_{i=1}^m \xi(z_i)}{\sqrt{\mu+\varepsilon}} \ge \alpha\sqrt{\varepsilon}\right\} \le \exp\left\{-\frac{3\alpha^2 m\varepsilon}{8M}\right\}.$$

Proof As ξ satisfies $|\xi - \mu| \le M$, the one-side Bernstein inequality tells us that

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\left\{\frac{\mu - \frac{1}{m}\sum_{i=1}^m \xi(z_i)}{\sqrt{\mu+\varepsilon}} \ge \alpha\sqrt{\varepsilon}\right\} \le \exp\left\{-\frac{\alpha^2 m(\mu+\varepsilon)\varepsilon}{2\left(\sigma^2 + \frac{1}{3}M\alpha\sqrt{\mu+\varepsilon}\sqrt{\varepsilon}\right)}\right\}.$$

Here $\sigma^2 \le E(\xi^2) \le ME(\xi) = M\mu$ since $0 \le \xi \le M$. Then we find that

$$\sigma^2 + \frac{1}{3}M\alpha\sqrt{\mu+\varepsilon}\sqrt{\varepsilon} \le M\mu + \frac{1}{3}M(\mu+\varepsilon) \le \frac{4}{3}M(\mu+\varepsilon).$$

This yields the desired inequality. ∎

Now we can turn to the error bound involving a function set. Lemma 17 yields the following bounds concerning the loss function V:

$$|\mathcal{E}_{\mathbf{z}}(f) - \mathcal{E}_{\mathbf{z}}(g)| \le q\max\left\{(1+\|f\|_\infty)^{q-1}, (1+\|g\|_\infty)^{q-1}\right\}\|f-g\|_\infty \tag{56}$$

and

$$|\mathcal{E}(f) - \mathcal{E}(g)| \le q\max\left\{(1+\|f\|_\infty)^{q-1}, (1+\|g\|_\infty)^{q-1}\right\}\|f-g\|_\infty. \tag{57}$$

Lemma 19 *Let $\mathcal{F}$ be a subset of $C(X)$ such that $\|f\|_\infty \le 1$ for each $f \in \mathcal{F}$. Then for every $\varepsilon > 0$ and $0 < \alpha \le 1$, we have*

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\left\{\sup_{f\in\mathcal{F}}\frac{\mathcal{E}(f) - \mathcal{E}_{\mathbf{z}}(f)}{\sqrt{\mathcal{E}(f)+\varepsilon}} \ge 4\alpha\sqrt{\varepsilon}\right\} \le \mathcal{N}\left(\mathcal{F}, \frac{\alpha\varepsilon}{q2^{q-1}}\right)\exp\left\{-\frac{3\alpha^2 m\varepsilon}{2^{q+3}}\right\}.$$

Proof Let $\{f_j\}_{j=1}^J \subset \mathcal{F}$ with $J = \mathcal{N}\left(\mathcal{F}, \frac{\alpha\varepsilon}{q2^{q-1}}\right)$ such that $\mathcal{F}$ is covered by balls centered at f_j with radius $\frac{\alpha\varepsilon}{q2^{q-1}}$. Note that $\xi = V(y, f(x))$ satisfies $0 \le \xi \le (1+\|f\|_\infty)^q \le 2^q$ for $f \in \mathcal{F}$. Then for each j, Lemma 18 tells

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\left\{\frac{\mathcal{E}(f_j)-\mathcal{E}_{\mathbf{z}}(f_j)}{\sqrt{\mathcal{E}(f_j)+\varepsilon}} \ge \alpha\sqrt{\varepsilon}\right\} \le \exp\left\{-\frac{3\alpha^2 m\varepsilon}{8\cdot 2^q}\right\}.$$

For each $f \in \mathcal{F}$, there is some j such that $\|f-f_j\|_\infty \le \frac{\alpha\varepsilon}{q2^{q-1}}$. This in connection with (56) and (57) tells us that $|\mathcal{E}_{\mathbf{z}}(f)-\mathcal{E}_{\mathbf{z}}(f_j)|$ and $|\mathcal{E}(f)-\mathcal{E}(f_j)|$ are both bounded by $\alpha\varepsilon$. Hence

$$\frac{|\mathcal{E}_{\mathbf{z}}(f)-\mathcal{E}_{\mathbf{z}}(f_j)|}{\sqrt{\mathcal{E}(f)+\varepsilon}} \le \alpha\sqrt{\varepsilon} \qquad \text{and} \qquad \frac{|\mathcal{E}(f)-\mathcal{E}(f_j)|}{\sqrt{\mathcal{E}(f)+\varepsilon}} \le \alpha\sqrt{\varepsilon}.$$

The latter implies that $\sqrt{\mathcal{E}(f_j)+\varepsilon} \le 2\sqrt{\mathcal{E}(f)+\varepsilon}$. Therefore,

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\left\{\sup_{f\in\mathcal{F}}\frac{\mathcal{E}(f)-\mathcal{E}_{\mathbf{z}}(f)}{\sqrt{\mathcal{E}(f)+\varepsilon}} \ge 4\alpha\sqrt{\varepsilon}\right\} \le \sum_{j=1}^J \mathrm{Prob}\left\{\frac{\mathcal{E}(f_j)-\mathcal{E}_{\mathbf{z}}(f_j)}{\sqrt{\mathcal{E}(f_j)+\varepsilon}} \ge \alpha\sqrt{\varepsilon}\right\}$$

which is bounded by $J\cdot\exp\left\{-\frac{3\alpha^2 m\varepsilon}{2^{q+3}}\right\}$. ■

Take $\mathcal{F}$ to be the set (55). The following covering number estimate will be used.

Lemma 20 *Let $\mathcal{F}$ be given by (55) with $R \ge \widetilde{\kappa}$ and $B = 1+\kappa R$. Its covering number in $C(X)$ can be bounded as follows:*

$$\mathcal{N}(\mathcal{F},\eta) \le \mathcal{N}\left(\frac{\eta}{2R}\right), \qquad \forall \eta > 0.$$

Proof It follows from the fact $\|\pi(f)-\pi(g)\|_\infty \le \|f-g\|_\infty$ that

$$\mathcal{N}(\mathcal{F},\eta) \le \mathcal{N}(B_R+[-B,B],\eta).$$

The latter is bounded by $\{(\kappa+\frac{1}{\widetilde{\kappa}})\frac{2R}{\eta}+1\}\mathcal{N}\left(B_R,\frac{\eta}{2}\right)$ since $\frac{2B}{\eta} \le (\kappa+\frac{1}{\widetilde{\kappa}})\frac{2R}{\eta}$ and

$$\|(f^*+b_f)-(g^*+b_g)\|_\infty \le \|f^*-g^*\|_\infty + |b_f-b_g|.$$

Note that an $\frac{\eta}{2R}$-covering of $\mathcal{B}$ is the same as an $\frac{\eta}{2}$-covering of B_R. Then our conclusion follows from Definition 6. ■

We are in a position to state our main result on the error analysis. Recall θ_M in (25).

Theorem 21 *Let $f_{K,C} \in \overline{\mathcal{H}_K}$, $M \ge \|V(y, f_{K,C}(x))\|_\infty$, and $0 < \beta \le 1$. For every $\varepsilon > 0$, we have*

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q) \le (1+\beta)(\varepsilon+\mathcal{D}(C))$$

with confidence at least $1-F(\varepsilon)$ where $F:\mathbb{R}_+\to\mathbb{R}_+$ is defined by

$$F(\varepsilon) := \exp\left\{-\frac{m\varepsilon^2}{2M(\Delta+\varepsilon)}\right\} + \mathcal{N}\left(\frac{\beta(\varepsilon+\mathcal{D}(C))^{3/2}}{q2^{q+3}\Sigma}\right)\exp\left\{-\frac{3m\beta^2(\mathcal{D}(C)+\varepsilon)^2}{2^{q+9}(\Delta+\varepsilon)}\right\} \tag{58}$$

with $\Delta := \mathcal{D}(C)+\mathcal{E}(f_q)$ and $\Sigma := \sqrt{2C(\Delta+\varepsilon)(\Delta+\theta_M\varepsilon)}$.

Proof Let $\varepsilon > 0$. We prove our conclusion in three steps.

Step 1: Estimate $\mathcal{E}_{\mathbf{z}}(f_{K,C}) - \mathcal{E}(f_{K,C})$.

Consider the random variable $\xi = V(y, f_{K,C}(x))$ with $0 \le \xi \le M$. If $M > 0$, since $\sigma^2(\xi) \le M\mathcal{E}(f_{K,C}) \le M(\mathcal{D}(C) + \mathcal{E}(f_q))$, by the one-side Bernstein inequality we obtain $\mathcal{E}_{\mathbf{z}}(f_{K,C}) - \mathcal{E}(f_{K,C}) \le \varepsilon$ with confidence at least

$$1 - \exp\left\{-\frac{m\varepsilon^2}{2(\sigma^2(\xi) + \frac{1}{3}M\varepsilon)}\right\} \ge 1 - \exp\left\{-\frac{m\varepsilon^2}{2M(\Delta+\varepsilon)}\right\}.$$

If $M = 0$, then $\xi = 0$ almost everywhere. Hence $\mathcal{E}_{\mathbf{z}}(f_{K,C}) - \mathcal{E}(f_{K,C}) = 0$ with probability 1. Thus, in both cases, there exists $U_1 \in Z^m$ with measure at least $1 - \exp\left\{-\frac{m\varepsilon^2}{2M(\varepsilon+\Delta)}\right\}$ such that $\mathcal{E}_{\mathbf{z}}(f_{K,C}) - \mathcal{E}(f_{K,C}) \le \theta_M\varepsilon$ whenever $\mathbf{z} \in U_1$.

Step 2: Estimate $\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}_{\mathbf{z}}(\pi(f_{\mathbf{z}}))$.

Let $\mathcal{F}$ be given by (55) with $R = \sqrt{2C(\Delta + \theta_M\varepsilon)}$ and $B = 1 + \kappa R$. By Proposition 4, $R \ge \sqrt{2C\mathcal{D}(C)} \ge \widetilde{\kappa}$. Applying Lemma 19 to $\mathcal{F}$ with $\tilde{\varepsilon} := \mathcal{D}(C) + \varepsilon > 0$ and $\alpha = \frac{\beta}{8}\sqrt{\tilde{\varepsilon}/(\tilde{\varepsilon} + \mathcal{E}(f_q))} \in (0, 1/8]$, we have

$$\mathcal{E}(f) - \mathcal{E}_{\mathbf{z}}(f) \le 4\alpha\sqrt{\tilde{\varepsilon}}\sqrt{\mathcal{E}(f) - \mathcal{E}(f_q) + \tilde{\varepsilon} + \mathcal{E}(f_q)}, \qquad \forall f \in \mathcal{F} \tag{59}$$

for $\mathbf{z} \in U_2$ where U_2 is a subset of Z^m with measure at least

$$1 - \mathcal{N}\Big(\mathcal{F}, \frac{\alpha\tilde{\varepsilon}}{q2^{q-1}}\Big)\exp\left\{-\frac{3m\alpha^2\tilde{\varepsilon}}{2^{q+3}}\right\} \ge 1 - \mathcal{N}\Big(\frac{\beta(\varepsilon + \mathcal{D}(C))^{3/2}}{q2^{q+3}\Sigma}\Big)\exp\left\{-\frac{3m\beta^2(\mathcal{D}(C)+\varepsilon)^2}{2^{q+9}(\Delta+\varepsilon)}\right\}.$$

In the above inequality we have used Lemma 20 to bound the covering number.

For $\mathbf{z} \in U_1 \cap U_2$, we have

$$\frac{1}{2C}\|f_{\mathbf{z}}^*\|_K^2 \le \mathcal{E}_{\mathbf{z}}(f_{K,C}) + \frac{1}{2C}\|f_{K,C}^*\|_K^2 \le \mathcal{E}(f_{K,C}) + \theta_M\varepsilon + \frac{1}{2C}\|f_{K,C}^*\|_K^2$$

which equals to $\mathcal{D}(C) + \theta_M\varepsilon + \mathcal{E}(f_q) = \Delta + \theta_M\varepsilon$. It follows that $\|f_{\mathbf{z}}^*\|_K \le R$. By Lemma 16, $|b_{\mathbf{z}}| \le B$. This means, $\pi(f_{\mathbf{z}}) \in \mathcal{F}$ and (59) is valid for $f = \pi(f_{\mathbf{z}})$.

Step 3: Bound $\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q)$ using (14).

Let α and $\tilde{\varepsilon}$ be the same as in Step 2. For $\mathbf{z} \in U_1 \bigcap U_2$, both $\mathcal{E}_{\mathbf{z}}(f_{K,C}) - \mathcal{E}(f_{K,C}) \le \theta_M\varepsilon \le \varepsilon$ and (59) with $f = \pi(f_{\mathbf{z}})$ hold true. Then (14) tells us that

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q) \le 4\alpha\sqrt{\tilde{\varepsilon}}\sqrt{(\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q)) + \tilde{\varepsilon} + \mathcal{E}(f_q)} + \tilde{\varepsilon}.$$

Denote $r := \mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q) + \tilde{\varepsilon} + \mathcal{E}(f_q) > 0$, we see that

$$r \le \tilde{\varepsilon} + \mathcal{E}(f_q) + 4\alpha\sqrt{\tilde{\varepsilon}}\sqrt{r} + \tilde{\varepsilon}.$$

It follows that

$$\sqrt{r} \le 2\alpha\sqrt{\tilde{\varepsilon}} + \sqrt{4\alpha^2\tilde{\varepsilon} + 2\tilde{\varepsilon} + \mathcal{E}(f_q)}.$$

Hence

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q) = r - (\tilde{\varepsilon} + \mathcal{E}(f_q)) \le \tilde{\varepsilon} + 8\alpha^2\tilde{\varepsilon} + 4\alpha\tilde{\varepsilon}\sqrt{4\alpha^2 + 2 + \mathcal{E}(f_q)/\tilde{\varepsilon}}.$$

Putting the choice of α into above, we find that

$$\mathcal{E}(\pi(f_{\mathbf{z}}))-\mathcal{E}(f_q)\le\tilde{\varepsilon}+\frac{\beta^2\tilde{\varepsilon}^2}{8\,(\tilde{\varepsilon}+\mathcal{E}(f_q))}+\frac{\beta\tilde{\varepsilon}}{2}\sqrt{\frac{\tilde{\varepsilon}}{\tilde{\varepsilon}+\mathcal{E}(f_q)}}\sqrt{\frac{\beta^2\tilde{\varepsilon}}{16\,(\tilde{\varepsilon}+\mathcal{E}(f_q))}+2+\frac{\mathcal{E}(f_q)}{\tilde{\varepsilon}}}$$

which is bounded by $(1+\beta)\tilde{\varepsilon}=(1+\beta)(\varepsilon+\mathcal{D}(C))$.

Finally, noting that the measure of $U_1\bigcap U_2$ is at least $1-F(\varepsilon)$, the proof is finished. ■

Observe that the function F is strictly decreasing and $F(0)>1$, $\lim_{\varepsilon\to\infty}F(\varepsilon)=0$. Hence for every $0<\delta<1$ there exists a unique number $\varepsilon>0$ satisfying $F(\varepsilon)=\delta$. Also, for a fixed $\varepsilon>0$, $F(\varepsilon)<\delta$ for sufficiently large m, since $F(\varepsilon)$ can be written as a^m+cb^m with $0<a,b<1$. Therefore, Theorem 21 can be restated in the following form.

Corollary 22 *Let F be given in Theorem 21. For every $0<\delta<1$, define $\varepsilon(\delta,m,C,\beta)>0$ to be the unique solution to the equation $F(\varepsilon)=\delta$. Then with confidence at least $1-\delta$, (20) holds. Moreover, $\lim_{m\to\infty}\varepsilon(\delta,m,C,\beta)=0$.*

Now we can prove the statements we made in Section 2 for the deterministic case.

Proof of Proposition 8. Take $\beta=1$. Since $\mathcal{R}(f_c)=0$, we know that $f_q=f_c$, $\mathcal{E}(f_q)=0$ and $\Delta=\mathcal{D}(C)$. Then $\Sigma\le\sqrt{2C(\mathcal{D}(C)+\theta_M\varepsilon)}\sqrt{\mathcal{D}(C)+\varepsilon}$ and

$$\mathcal{N}\left(\frac{\beta(\varepsilon+\mathcal{D}(C))^{3/2}}{q2^{q+3}\Sigma}\right)\le\mathcal{N}\left(\frac{\varepsilon}{q2^{q+3}\mathcal{M}}\right).$$

The function value $F(\varepsilon)$ can be bounded by

$$\exp\left\{-\frac{m\varepsilon^2}{2M(\mathcal{D}(C)+\varepsilon)}\right\}+\mathcal{N}\left(\frac{\varepsilon}{q2^{q+3}\mathcal{M}}\right)\exp\left\{-\frac{3m\varepsilon}{2^{q+9}}\right\}.$$

The first term above is bounded by $\delta/2$ when $\varepsilon\ge\frac{4M\log(2/\delta)}{m}+\mathcal{D}(C)$. The second term is at most $\delta/2$ if $\varepsilon\ge\varepsilon^*$. Therefore

$$\varepsilon(\delta,m,C,\beta)\le\max\left\{\frac{4M\log(2/\delta)}{m}+\mathcal{D}(C),\varepsilon^*\right\},$$

and (26) follows from Theorem 21.

(a) When (23) holds, noting that the left side of (27) is strictly decreasing, it is easy to check that

$$\varepsilon^*\le 2^{q+9}\left\{1+c((q+4)\log 8)^s\right\}\frac{(\log m+\log(C\mathcal{D}(C))+\log(C\theta_M))^s+\log(2/\delta)}{m}.$$

This yields the stated bound for ε^* in the case (a).

(b) If (24) is true, then the function defined by

$$\widetilde{F}(\varepsilon):=c\frac{(q2^{q+4})^s\{(2C\mathcal{D}(C))^{s/2}+(2C\varepsilon)^{s/2}\}}{\varepsilon^s}-\frac{3m\varepsilon}{2^{q+9}}$$

satisfies $\widetilde{F}(\varepsilon^*) \geq \log(\delta/2)$. Since $\widetilde{F}$ is a decreasing function, $\varepsilon^* \leq \widetilde{\varepsilon}^*$ whenever $\widetilde{F}(\widetilde{\varepsilon}^*) \leq \log(\delta/2)$.

If $\widetilde{\varepsilon}^* \geq rm^{-1/(s+1)}(2C\mathcal{D}(C))^{s/(2s+2)}\log\frac{2}{\delta}$ with

$$r \geq 2^{q+8}\widetilde{\kappa}^{-s/(s+1)} + q2^{q+10}c^{1/(s+1)}/\log 2, \tag{60}$$

then

$$c(q2^{q+4})^s\frac{(2C\mathcal{D}(C))^{s/2}}{(\widetilde{\varepsilon}^*)^s} - \frac{m\widetilde{\varepsilon}^*}{2^{q+8}} \leq \frac{rm^{s/(s+1)}}{2^{q+8}}(2C\mathcal{D}(C))^{\frac{s}{2s+2}}\log\frac{2}{\delta}\left\{\frac{c(q2^{q+4})^s 2^{q+8}}{(r\log(2/\delta))^{s+1}} - 1\right\}.$$

Since $r \geq q2^{q+10}c^{1/(s+1)}/\log 2$, according to Proposition 4, this can be bounded by

$$\frac{r}{2^{q+8}}m^{s/(s+1)}(2C\mathcal{D}(C))^{s/(2s+2)}\log\frac{2}{\delta}\left\{-\frac{1}{2}\right\} \leq \frac{r}{2^{q+9}}\widetilde{\kappa}^{s/(s+1)}\log\frac{\delta}{2} \leq \frac{1}{2}\log\frac{\delta}{2}.$$

Here we have used the condition $r \geq 2^{q+8}\widetilde{\kappa}^{-s/(s+1)}$.

In the same way, if $\widetilde{\varepsilon}^* \geq rm^{-2/(s+2)}(2C)^{s/(s+2)}\log\frac{2}{\delta}$ with

$$r \geq 2^{q+9} + q^2 4^{q+5}c^{2/(s+2)}/\log 2, \tag{61}$$

we have for $C \geq 1/2$,

$$c(q2^{q+4})^s\left(\frac{2C}{\widetilde{\varepsilon}^*}\right)^{s/2} - \frac{m\widetilde{\varepsilon}^*}{2^{q+9}} \leq \frac{1}{2}\log\frac{\delta}{2}.$$

Combining the above two bounds, we obtain the desired estimate (28) with $\widetilde{c}$ determined by the two conditions (60) and (61). The proof of Proposition 8 is complete. ■

In the general case, the following bounds hold.

Corollary 23 *For every $0 < \delta \leq 1$, with confidence at least $1-\delta$ there holds*

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_q) \leq 2\max\left\{\frac{2^{q+1}\left(1+\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)}\right)^{q/2}\sqrt{\log(2/\delta)}}{\sqrt{m}}, \varepsilon^*\right\} + 2\widetilde{\mathcal{D}}(C),$$

where ε^* is the solution to the equation

$$\log\mathcal{N}\left(\frac{\varepsilon^{3/2}}{q4^{q+2}\sqrt{2C}}\right) - \frac{3m\varepsilon^2}{4^{q+5}} = \log\frac{\delta}{2}. \tag{62}$$

Proof Take $\beta = 1$ and $f_{K,C} = \widetilde{f}_{K,C}$. By Lemma 16, $\|\widetilde{f}_{K,C}\|_\infty \leq 1 + 2\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)}$ and we can take $M = 2^q(1+\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)})^q$. Also, $\Delta = \widetilde{\mathcal{D}}(C) + \mathcal{E}(f_q) \leq 1$. It follows that $\Sigma \leq \sqrt{2C}(\Delta+\varepsilon) \leq \sqrt{2C}(1+\varepsilon)$.

Since $\mathcal{E}(\pi(f_{\mathbf{z}})) \leq 2^q$, we only need to consider the range $\varepsilon \leq 2^q$ to bound $\varepsilon(\delta, m, C, \beta)$. In this range,

$$\mathcal{N}\left(\frac{\beta(\varepsilon+\widetilde{\mathcal{D}}(C))^{3/2}}{q2^{q+3}\Sigma}\right) \leq \mathcal{N}\left(\frac{\varepsilon^{3/2}}{q4^{q+2}\sqrt{2C}}\right).$$

Then $F(\varepsilon)$ can be bounded by

$$\exp\left\{-\frac{m\varepsilon^2}{4^{q+1}(1+\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)})^q}\right\}+\mathcal{N}\left(\frac{\varepsilon^{3/2}}{q4^{q+2}\sqrt{2C}}\right)\exp\left\{-\frac{3m\varepsilon^2}{4^{q+5}}\right\}.$$

Thus $F(\varepsilon)\le\delta$ if

$$\varepsilon\ge\max\left\{\frac{2^{q+1}(1+\kappa\sqrt{2C\widetilde{\mathcal{D}}(C)})^{q/2}\sqrt{\log(2/\delta)}}{\sqrt{m}},\varepsilon^*\right\}$$

where ε^* is the solution to the equation (62). This together with Theorem 21 yields the desired estimate. ■

The bound for the sample error derived in Corollary 23 may be further improved by the well developed empirical process techniques in the literature. We shall discuss this elsewhere.

The total error (14) consists of two parts. We shall not discuss the possibility of further improving the sample error bound here, because it is of the same importance to understand the regularization error. This becomes more important when not much estimate is available for the regularization error. In the previous sections, we could compute $\mathcal{D}(C)$ explicitly only for special cases. Most of the time, ρ is not strictly separable, even not weakly separable. Hence it is desirable to estimate $\mathcal{D}(C)$ explicitly for general distributions. In the following we shall choose $f_{K,C}=\widetilde{f}_{K,C}$ and estimate $\widetilde{\mathcal{D}}(C)$.

6. Error Analysis by Approximation in L^q Spaces

The main result on the convergence analysis given in Section 5 enables us to have some nice observations. These follow from facts on approximation in L^q spaces.

Lemma 24 *If* $1<q\le 2$, *then*

$$(1+u)_+^q\le 1+|u|^q+qu,\qquad\forall u\in\mathbb{R}.$$

Proof Set the continuous function $f(u):=1+|u|^q+qu-(1+u)_+^q$. Then $f(0)=0$.

Since $0<q-1\le 1$, for $u>0$ we have

$$f'(u)=q\left(1+u^{q-1}-(1+u)^{q-1}\right)\ge 0.$$

Hence $f(u)\ge 0$ for $u\ge 0$.

For $-1<u<0$, we see that

$$f'(u)=q\left(1-(-u)^{q-1}-(1+u)^{q-1}\right)=q\left(1-|u|^{q-1}-(1-|u|)^{q-1}\right)\le 0.$$

Hence $f(u)\ge 0$ for $-1<u<0$.

Finally, when $u<-1$, there holds

$$f'(u)=q\left(1-(-u)^{q-1}\right)\le 0.$$

Therefore, we also have $f(u)\ge f(-1)\ge 0$ for $u\le -1$.

Thus $f(u)\ge 0$ on the whole real line and Lemma 24 is proved. ■

Theorem 25 *Let $f : X \to \mathbb{R}$ be measurable. Then*

$$\mathcal{E}(f) - \mathcal{E}(f_q) \le \begin{cases} \|f - f_q\|^q_{L^q_{\rho_X}}, & \text{if } 1 < q \le 2, \\ q2^{q-1}\|f - f_q\|_{L^q_{\rho_X}} \left(2^{q-1} + \|f - f_q\|^{q-1}_{L^q_{\rho_X}}\right), & \text{if } q > 2. \end{cases}$$

Proof Since $|f_q(x)| \le 1$, by the second inequality of Lemma 17, for each $x \in X$ we have

$$\begin{aligned} \mathcal{E}(f|x) - \mathcal{E}(f_q|x) &= \int_Y (1 - yf(x))^q_+ - (1 - yf_q(x))^q_+ d\rho(y|x) \\ &\le q4^{q-1}|f(x) - f_q(x)| + q2^{q-1}|f(x) - f_q(x)|^q. \end{aligned}$$

It follows that

$$\mathcal{E}(f) - \mathcal{E}(f_q) = \int_X \mathcal{E}(f|x) - \mathcal{E}(f_q|x) d\rho_X \le q4^{q-1}\|f - f_q\|_{L^1_{\rho_X}} + q2^{q-1}\|f - f_q\|^q_{L^q_{\rho_X}}.$$

Then the inequality for the case $q > 2$ follows from the Hölder inequality.

Turn to the case $1 < q \le 2$. It is sufficient to show that for each $x \in X$,

$$\mathcal{E}(f|x) - \mathcal{E}(f_q|x) \le |f(x) - f_q(x)|^q. \tag{63}$$

The definition (10) of f_q tells us that

$$(1 + f_q(x))^{q-1} = \frac{2^{q-1}(1 + f_\rho(x))}{\{(1 + f_\rho(x))^{1/(q-1)} + (1 - f_\rho(x))^{1/(q-1)}\}^{q-1}}$$

and

$$(1 - f_q(x))^{q-1} = \frac{2^{q-1}(1 - f_\rho(x))}{\{(1 + f_\rho(x))^{1/(q-1)} + (1 - f_\rho(x))^{1/(q-1)}\}^{q-1}}.$$

These expressions in connection with (45) imply

$$\mathcal{E}(f|x) = \frac{(1 + f(x))^q_+(1 - f_q(x))^{q-1}}{(1 + f_q(x))^{q-1} + (1 - f_q(x))^{q-1}} + \frac{(1 - f(x))^q_+(1 + f_q(x))^{q-1}}{(1 + f_q(x))^{q-1} + (1 - f_q(x))^{q-1}}$$

and together with (48)

$$\mathcal{E}(f_q|x) = \frac{2(1 - f_q(x))^{q-1}(1 + f_q(x))^{q-1}}{(1 + f_q(x))^{q-1} + (1 - f_q(x))^{q-1}}.$$

Thus (63) follows from the following inequality (by taking $t = f(x)$ and $\theta = f_q(x)$):

$$\begin{aligned} &\frac{(1+t)^q_+(1-\theta)^{q-1}}{(1+\theta)^{q-1} + (1-\theta)^{q-1}} + \frac{(1-t)^q_+(1+\theta)^{q-1}}{(1+\theta)^{q-1} + (1-\theta)^{q-1}} \\ &\qquad -2\frac{(1-\theta)^{q-1}(1+\theta)^{q-1}}{(1+\theta)^{q-1} + (1-\theta)^{q-1}} \le |t - \theta|^q, \qquad \forall t \in \mathbb{R}, \theta \in (-1, 1). \end{aligned} \tag{64}$$

What is left is to verify the inequality (64). Since $-1 < \theta < 1$, we have

$$(1+t)^q_+(1-\theta)^{q-1} = (1-\theta^2)^{q-1}(1+\theta)\left(\frac{1+t}{1+\theta}\right)^q_+ = (1-\theta^2)^{q-1}(1+\theta)\left(1 + \frac{t-\theta}{1+\theta}\right)^q_+.$$

By Lemma 24 with $u=(t-\theta)/(1+\theta)$, we see that

$$(1+t)_+^q(1-\theta)^{q-1}\le(1-\theta^2)^{q-1}(1+\theta)\left(1+\left|\frac{t-\theta}{1+\theta}\right|^q+q\frac{t-\theta}{1+\theta}\right).$$

In the same way, by Lemma 24 with $u=-(t-\theta)/(1-\theta)$, we have

$$(1-t)_+^q(1+\theta)^{q-1}\le(1-\theta^2)^{q-1}(1-\theta)\left(1+\left|\frac{t-\theta}{1-\theta}\right|^q-q\frac{t-\theta}{1-\theta}\right).$$

Combining the above two estimates, we obtain

$$(1+t)_+^q(1-\theta)^{q-1}+(1-t)_+^q(1+\theta)^{q-1}\le 2(1-\theta^2)^{q-1}+|t-\theta|^q\left\{(1-\theta)^{q-1}+(1+\theta)^{q-1}\right\}.$$

This proves our claim (64), thereby Theorem 25. ■

Recall the K-functional given by (19).

Theorem 26 *For each $C>0$, there holds*

$$\widetilde{\mathcal{D}}(C)\le\mathcal{K}(f_q,\frac{1}{2C}).$$

Proof The case $1<q\le 2$ is an easy consequence of Theorem 25.

Turn to the case $q>2$. The special choice $f=0+0\in\overline{\mathcal{H}}_K$ and the fact $\|f_q\|_{L^q_{\rho_X}}\le 1$ tell us that for any $t>0$,

$$\mathcal{K}(f_q,t)=\inf_{\substack{f\in\overline{\mathcal{H}}_K\\ \|f-f_q\|_{L^q_{\rho_X}}\le 1}}\left\{q2^{q-1}(2^{q-1}+1)\|f-f_q\|_{L^q_{\rho_X}}+t\|f^*\|_K^2\right\}.$$

According to Theorem 25, for $f\in\overline{\mathcal{H}}_K$ with $\|f-f_q\|_{L^q_{\rho_X}}\le 1$, we have

$$\mathcal{E}(f)-\mathcal{E}(f_q)\le q2^{q-1}(2^{q-1}+1)\|f-f_q\|_{L^q_{\rho_X}}.$$

Thus,

$$\widetilde{\mathcal{D}}(C)=\inf_{f\in\overline{\mathcal{H}}_K}\left\{\mathcal{E}(f)-\mathcal{E}(f_q)+\frac{1}{2C}\|f^*\|_K^2\right\}\le\mathcal{K}(f_q,\frac{1}{2C})$$

and the proof of Theorem 26 is complete. ■

We can now derive several observations from Theorem 26.

The first observation says that the SVM q-classifier converges when f_q lies in the closure of $\overline{\mathcal{H}}_K$ in $L^q_{\rho_X}$. In particular, for any Borel probability measure ρ, this is always the case if K is a universal kernel since $C(X)$ is dense in $L^q_{\rho_X}$.

Corollary 27 *If f_q lies in the closure of $\overline{\mathcal{H}}_K$ in $L^q_{\rho_X}$, then for every $\varepsilon>0$ and $0<\delta<1$, there exist $C_\varepsilon>0$, $m_0\in\mathbb{N}$ and a sequence C_m with $\lim_{m\to\infty}C_m=\infty$ such that*

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\left\{\mathcal{R}(sgn(f_{\mathbf{z}}))-\mathcal{R}(f_c)\le\varepsilon\right\}\ge 1-\delta,\qquad\forall m\ge m_0,\ C_\varepsilon\le C\le C_m.$$

Proof Since f_q lies in the closure of $\overline{\mathcal{H}_K}$ in $L^q_{\rho_X}$, for every $\varepsilon > 0$ there is some $f_\varepsilon = f^*_\varepsilon + b_\varepsilon \in \overline{\mathcal{H}_K}$ such that $q2^q(2^{q-1}+1)\|f_\varepsilon - f_q\|_{L^q_{\rho_X}} \le \varepsilon^2/16$. Take $C_\varepsilon = 8\|f^*_\varepsilon\|^2_K/\varepsilon^2$. Then for any $C \ge C_\varepsilon$, we have $\frac{1}{2C}\|f^*_\varepsilon\|^2_K \le \varepsilon^2/16$. Theorem 26 tells us that $\widetilde{\mathcal{D}}(C) \le \varepsilon^2/8$.

Take m_0 such that

$$\frac{2^{q+1}(1+\kappa\sqrt{2C_\varepsilon})^{q/2}\sqrt{\log(2/\delta)}}{\sqrt{m_0}} \le \frac{\varepsilon^2}{8}$$

and

$$\log \mathcal{N}\left(\frac{\varepsilon^3}{q4^{q+4}\sqrt{2C_\varepsilon}}\right) - \frac{3m_0\varepsilon^4}{4^{q+8}} \le \log\frac{\delta}{2}.$$

Also, we choose C_m such that

$$\frac{2^{q+1}(1+\kappa\sqrt{2C_m})^{q/2}\sqrt{\log(2/\delta)}}{\sqrt{m}} < \frac{\varepsilon^2}{8}$$

and

$$\log \mathcal{N}\left(\frac{\varepsilon^3}{q4^{q+4}\sqrt{2C_m}}\right) - \frac{3m\varepsilon^4}{4^{q+8}} \le \log\frac{\delta}{2}.$$

Then by Corollary 5.2, $\varepsilon(\delta, m, C, \beta) \le \frac{\varepsilon^2}{8}$ when $m \ge m_0$ and $C_\varepsilon \le C \le C_m$. Together with Theorem 14, our conclusion is proved. ■

Our second observation from Theorem 26 concerns nonuniversal kernels which nonetheless ensures the convergence of the SVM q-classifier. The point here is that $\mathcal{H}_K$ is not dense in $C(X)$, but after adding the offset the space $\overline{\mathcal{H}_K}$ becomes dense.

Example 2 *Let K be a Mercer kernel on $X = [0,1]$:*

$$K(x,y) = \sum_{j\in J} a_j (x\cdot y)^j,$$

where J is a subset of $\mathbb{N}$, $a_j > 0$ for each $j \in J$, and $\sum_{j\in J} a_j < \infty$. Note that this kernel satisfies $K_0(y) \equiv 0$, hence $f(0) = 0$ for all $f \in \mathcal{H}_K$. Hence the space $\mathcal{H}_K$ is not dense in $C(X)$ and K is not an universal kernel. But if $\sum_{j\in J} \frac{1}{j} = \infty$, then $\overline{\mathcal{H}_K}$ is dense in $C(X)$ (Zhou 2003a) and hence in $L^q_{\rho_X}$. Therefore, the SVM q-classifier associated with the (identical) kernel K converges.

Remark 28 *In Section 4 and the proof of Theorem 21, we have shown how the offset influences the sample error. Proposition 4 and Example 2 tell that it may also influence the approximation error. However, our analysis in the following two sections will not focus on this point and it may be an interesting topic.*

In practical applications, one can use varying kernels for (3).

Definition 29 *Let $\{K_d\}_{d\in\mathbb{N}}$ be a sequence of Mercer kernels on X. We say that the SVM q-classifier associated with the kernels $\{K_d\}$ converges if for a sequence $\{C_m\}_{m\in\mathbb{N}}$ of positive numbers, $f_{\mathbf{z}}$ defined by (3) with $K = K_d$ and $C = C_m$ satisfies the following:*

For every Borel probability measure ρ *on* Z*, and* $0 < \delta < 1$*,* $\varepsilon > 0$*, for sufficiently large* d *there is some* $m_d \in \mathbb{N}$ *such that*

$$\mathrm{Prob}_{\mathbf{z}\in Z^m}\{\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c) \le \varepsilon\} \ge 1-\delta, \qquad \forall m \ge m_d.$$

For a universal kernel K, one may take K_d to be identically K and the convergence holds. But the kernels could change such as the polynomial kernels (Boser, Guyon and Vapnik, 1992). Our third observation from Theorem 26 is to confirm the convergence of the SVM q-classifiers with these kernels.

Proposition 30 *For any* $1 < q < \infty, n \in \mathbb{N}$ *and* $X \subset \mathbb{R}^n$*, the SVM* q*-classifier associated with* $\{K_d\}_{d=1}^{\infty}$*, the polynomial kernels* $K_d(x,y) = (1+x\cdot y)^d$*, converges.*

Proposition 30 is a consequence of a quantitative result below.

Let P_d be the space of all polynomials of degree at most d. It is a RKHS $\mathcal{H}_{K_d}$ with the Mercer kernel $K_d(x,y) = (1+x\cdot y)^d$. The rich knowledge from approximation theory tells us that for an arbitrary Borel probability measure on Z, there is a sequence of polynomials $\{p_d \in P_d\}_{d=1}^{\infty}$ such that $\lim_{d\to\infty} \|f_q - p_d\|_{L^q_{\rho_X}} = 0$. The rate of this convergence depends on the regularity of the function f_q (hence the function f_ρ) and the marginal distribution ρ_X. With this in hand, we can now state the result on the convergence of the q-classifier with polynomial kernels K_d.

Corollary 31 *Let* $X \subset \mathbb{R}^n$*, and* ρ *be an arbitrary Borel probability measure on* Z*. Let* $d \in \mathbb{N}$ *and* $K_d(x,y) = (1+x\cdot y)^d$ *be the polynomial kernel. Set* $\|X\| := \sup_{x\in X}|x|$*. Let* $\{p_d \in P_d\}_{d=1}^{\infty}$ *satisfy* $E_d := \|f_q - p_d\|_{L^q_{\rho_X}} \to 0$ *(as* $d\to\infty$*). Set* $N := (n+d)!/(n!d!)+1$ *and* $0 < \sigma < \frac{2}{q}$*. Then there exists* $m_{q,\sigma} \in \mathbb{N}$ *such that for* $m \ge m_{q,\sigma}$ *and* $C = m^\sigma$*, for every* $0 < \delta < 1$*, with confidence* $1-\delta$ *there holds*

$$\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c) \le \sqrt{q}\,2^{q+1}\sqrt{E_d} + \frac{\|p_d\|_{K_d}}{m^{\sigma/2}} + \frac{2^{q+5}(N\log(\frac{2}{\delta}))^{1/4}(1+\|X\|^2)^{qd/8}}{m^{\frac{1}{4}-\frac{q\sigma}{8}}}.$$

Proof Take $p_d^* = p_d$ with zero offset in the K-functional. Then by Theorem 26,

$$\widetilde{\mathcal{D}}(C) \le \mathcal{K}(f_q, \frac{1}{2C}) \le q2^{q-1}(2^{q-1}+1)E_d + \frac{1}{2C}\|p_d\|^2_{K_d}.$$

The covering numbers of the finite dimensional space P_d (e.g. Cucker and Zhou, 2004) and (22) give us the estimate:

$$\mathcal{N}(\eta) \le (\kappa + \frac{1}{\tilde{\kappa}})\left(\frac{2}{\eta}+1\right)^N,$$

where $N-1 = (n+d)!/(n!d!)$ is the dimension of the space $P_d(\mathbb{R}^n)$. Also, $\kappa = \sqrt{\|K_d\|_\infty} \le (1+\|X\|^2)^{d/2}$.

Take $C = m^\sigma$. By Corollary 23, solving the equation (62) yields

$$\varepsilon(\delta,m,C,\beta) \le \frac{(2^{q+5}\sqrt{N} + \sqrt{\log(\kappa+\frac{1}{\tilde{\kappa}})}(\log m + \log(2/\delta))^{1/2}}{\sqrt{m}} + \frac{4^{q+2}\kappa^{\frac{q}{2}}\sqrt{\log(\frac{2}{\delta})}}{m^{\frac{1}{2}-\frac{q\sigma}{4}}}$$

which is bounded by $4^{5+q}\sqrt{N\log(\frac{2}{\delta})}\kappa^{q/2}m^{\frac{q\sigma}{4}-\frac{1}{2}}$ for $m \geq m_{q,\sigma}$. Here $m_{q,\sigma}$ is an integer depending on q,σ (but not on m, d or δ). Then for each $m \geq m_{q,\sigma}$, with confidence $1-\delta$ the desired estimate holds true. ■

Remark 32 *Note that P_d is a finite dimension space. Thus the norms $\|\cdot\|_{K_d}$ and $\|\cdot\|_{L^q_{\rho_X}}$ are equivalent for a fixed d when ρ_X is non-degenerate. It would be interesting to compare the norm $\|p\|_{K_d}$ with $\|p\|_{L^q_{\rho_X}}$ for $p \in P_d$ as d tends to infinity.*

Proof of Proposition 30. For every $\varepsilon > 0$, there exists some $d_0 \in \mathbb{N}$ such that $\sqrt{q}2^{q-1}\sqrt{E_d} \leq \varepsilon/2$ for every $d \geq d_0$. Then by Corollary 31 we can find some $m_{q,\sigma} \leq m_d \in \mathbb{N}$ such that $m_d^{-\sigma/2}\|p_d\|_{K_d} \leq \varepsilon/4$ and $2^{q+5}(N\log(2/\delta))^{1/4}(1+\|X\|^2)^{dq/8}m_d^{\frac{q\sigma}{8}-\frac{1}{4}} \leq \varepsilon/4$. Then for any $m \geq m_d$ we have $\mathcal{R}(\mathrm{sgn}(f_{\mathbf{z}})) - \mathcal{R}(f_c) \leq \varepsilon$ with confidence $1-\delta$. This proves Proposition 30. ■

7. Rate of Convergence for the q-Norm Soft Margin Classifier

Corollary 23 and Theorem 26 enable us to get the convergence rate for the SVM q-classifier. The rate depends on the K-functional $\mathcal{K}(f_q,t)$. It can be characterized by the quantity (Smale and Zhou 2003; Zhou 2003b)

$$I_q(g,R) := \inf_{\substack{f\in\mathcal{H}_K \\ \|f^*\|_K\leq R}} \left\{\|g-f\|_{L^q_{\rho_X}}\right\}. \tag{65}$$

Define

$$J_q(f_q,R) := \begin{cases} (I_q(f_q,R))^q, & \text{if } 1 < q \leq 2, \\ q2^{q-1}(2^{q-1}+1)I_q(f_q,R), & \text{if } q > 2. \end{cases}$$

Then the following corollary holds true.

Corollary 33 *For any $t > 0$ there holds*

$$\mathcal{K}(f_q,t) \leq \inf_{R>0}\{J_q(f_q,R)+tR^2\}.$$

One may choose appropriate R to estimate the convergence rate of $\mathcal{K}(f_q,t)$, which together with Corollary 23 gives the convergence rate of the V-risk and a strategy of choosing the regularization parameter C. In general, the choice of R depends on the regularity of f_q and the kernel K. Let us demonstrate this by examples.

In what follows let $X \subset \mathbb{R}^n$ have Lipschitz boundary and ρ be a probability measure such that $d\rho_X = dx$ is the Lebesgue measure. Consider $q = 2$ and thus $f_q = f_\rho$. We use the approximation error studied in Smale and Zhou (2003) (see also Niyogi and Girosi (1996) for related discussion):

$$I_2^*(g,R) := \inf_{\substack{f^*\in\mathcal{H}_K \\ \|f^*\|_K\leq R}} \{\|g-f^*\|_{L^2}\}$$

to bound the term $I_2(f_\rho,R)$. With the choice $b_f = 0$ we obtain

$$I_2(f_\rho,R) \leq I_2^*(f_\rho,R).$$

Note that we disregard the influence of the offset here and thus the rate need not be optimal.

The first example includes spline kernels (Wahba 1990).

Example 3 *Let $X \subset \mathbb{R}^n$ and K be a Mercer kernel such that $\mathcal{H}_K$ is the Sobolev space $H^r(X)$ with $r > n/2$. If f_ρ lies in the Sobolev space $H^s(X)$ with $0 < s < r$, then*

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_\rho) = O\left(\frac{\sqrt{\log m + C}}{\sqrt{m}} + \frac{C^{n/(4r+3n)}}{m^{2r/(4r+3n)}}\right) + O\left(C^{-s/r}\right).$$

Thus, C should be chosen such that $C \to \infty$, $C/m \to 0$ as $m \to \infty$.

Proof It was shown in Smale and Zhou (2003, Theorem 3.1) that for $0 < \theta < 1$, $I_2^*(f_\rho, R) = O(R^{-\theta/(1-\theta)})$ if and only if f_ρ lies in the interpolation space $(L^2_{\rho_X}, \mathcal{H}_K)_{\theta,\infty}$. It is well known that $H^s(X) \subset (L^2, H^r(X))_{s/r,\infty}$ for $0 < s < r$. Here $\mathcal{H}_K = H^r(X)$ and $d\rho_X = dx$. Therefore, the assumption $f_\rho \in H^s(X)$ tells us that $f_\rho \in (L^2_{\rho_X}, \mathcal{H}_K)_{s/r,\infty}$. Hence there holds

$$I_2(f_\rho, R) \le I_2^*(f_\rho, R) \le C_\rho R^{-s/(r-s)}$$

for some constant C_ρ. Choose $R = C_\rho^{(r-s)/r} C^{(r-s)/2r}$ to obtain

$$\mathcal{K}(f_\rho, \frac{1}{2C}) \le (I_2(f_\rho, R))^2 + \frac{R^2}{2C} \le \frac{3}{2} C_\rho^{2(r-s)/r} C^{-s/r}.$$

Using the well known covering number estimates for Sobolev spaces

$$\log \mathcal{N}(B_R, \eta) \le C_r \left(\frac{1}{\eta}\right)^{n/r}$$

and solving the equation (62), we see that

$$\varepsilon^* \le 2^6 \sqrt{\frac{\log \kappa + \log m + \log C + \log(2/\delta)}{m}} + 2^{6+3n/(4r)} \sqrt{C_r} \frac{C^{n/(4r+3n)}}{m^{2r/(4r+3n)}}.$$

This proves the conclusion. ■

Example 4 *Let $\sigma > 0, s > 0$ and K be the Gaussian kernel $K(x,y) = \exp\left\{-\frac{|x-y|^2}{\sigma^2}\right\}$.*

(a) *If f_ρ lies in the interpolation space $(L^2_{\rho_X}, \mathcal{H}_K)_{\theta,\infty}$ for some $0 < \theta < 1$, that is, $\mathcal{K}(f_\rho, t) \le C_\theta t^\theta$ for some constant C_θ, then for any $0 < \delta < 1$, with confidence $1 - \delta$, there holds*

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_\rho) = O\left(\frac{\sqrt{C + (\log m)^{n+1}}}{\sqrt{m}}\right) + O\left(C^{-\theta}\right).$$

This implies the parameter C should be taken to satisfy $C \to \infty$ and $C/m \to 0$ as $m \to \infty$. An asymptotic optimal choice is $C = O(m^{1/(1+2\theta)})$. With this choice, the convergence rate is $O(m^{-\theta/(1+2\theta)})$.

(b) *If $f_\rho \in H^s(X)$ with $s > 0$, then for any $0 < \delta < 1$, with confidence $1 - \delta$, we have*

$$\mathcal{E}(\pi(f_{\mathbf{z}})) - \mathcal{E}(f_\rho) = O\left(\frac{\sqrt{C + (\log m)^{n+1}}}{\sqrt{m}}\right) + O\left((\log C)^{-s/2}\right).$$

An asymptotically optimal choice is $C = O\left(\frac{m}{(\log m)^s}\right)$ which gives the convergence rate $O((\log m)^{-s/2})$.

Proof Solving the equation (62) with the covering number estimate (23) yields

$$\varepsilon^* = O\left(\frac{\sqrt{(1+c)(\log C + \log m)^{n+1}}}{\sqrt{m}}\right).$$

Then the statement in (a) follows easily from Corollary 23, Theorem 21 and Corollary 33.

To see the conclusion in (b), we notice that the assumption $f_\rho \in H^s(X)$ provides the approximation error estimates (Smale and Zhou, 2003; Zhou, 2003b)

$$I_2(f_\rho, R) \le I_2^*(f_\rho, R) \le C_s (\log R)^{-s/4}$$

for every $R > C_s$, where C_s is a constant depending on s, σ, n and the Sobolev norm of f_ρ. Choose $R = \sqrt{2C}(\log C)^{-s/4}$ to obtain

$$\mathcal{K}(f_\rho, \frac{1}{2C}) \le (I_2(f_\rho, R))^2 + \frac{R^2}{2C} \le (2^s C_s^2 + 1)(\log C)^{-s/2}.$$

Then the desired bound follows from Theorem 26 and the above established bound for ε^*. ■

It was shown in Smale and Zhou (2003) that for the Gaussian kernel in Example 4, $I_2^*(g, R) = O(R^{-\varepsilon})$ with $\varepsilon > 0$ only if g is C^∞. Hence logarithmic convergence rate is expected for a Sobolev function f_ρ. However, in practice, one often chooses the different variances σ of the Gaussian kernel according to the different sample size m. With this flexibility, the regularization error can be improved greatly and polynomial convergence rates are possible.

Acknowledgments

The authors would like to thank Professor Zhen Wang for his helpful discussion on the best constants in Theorem 14 and Theorem 25. They are also grateful to the referees for the constructive suggestions and comments.

This work is supported partially by the Research Grants Council of Hong Kong [Project No. CityU 1087/02P] and by City University of Hong Kong [Project No. 7001442]. The corresponding author is Ding-Xuan Zhou. Yiming Ying is on leave from Institute of Mathematics, Chinese Academy of Science, Beijing 100080, P. R. CHINA.

Appendix A. Error Comparison for a General Loss Function

In this appendix for a general convex loss function we bound the excess misclassification error by the excess V-risk. Here the loss function takes the form

$$V(y, f(x)) = \phi(y f(x))$$

for a univariate function $\phi : \mathbb{R} \to \mathbb{R}_+$.

For each $x \in X$, we denote

$$\mathcal{E}(f|x) := \int_Y V(y, f(x)) d\rho(y|x), \tag{66}$$

then $\mathcal{E}(f|x) = Q(\eta(x), f(x))$. Here $Q : [0,1] \times (\mathbb{R} \cup \{\pm\infty\}) \to \mathbb{R}_+$ is given by

$$Q(\eta, f) = \eta\phi(f) + (1-\eta)\phi(-f),$$

and $\eta : X \to \mathbb{R}$ is defined by $\eta(x) := P(\mathcal{Y} = 1|x)$. Set

$$f_\phi^*(\eta) := \arg \min_{f \in \mathbb{R} \cup \{\pm\infty\}} Q(\eta, f).$$

Then $f_\rho^V(x) = f_\phi^*(\eta(x))$. The main result of Zhang (2004) can be stated as follows.

Theorem A *Let ϕ be convex. Assume $f_\phi^*(\eta) > 0$ when $\eta > 0.5$. Assume there exists $c > 0$ and $s \geq 1$ such that for all $\eta \in [0,1]$,*

$$|0.5 - \eta|^s \leq c^s \left(Q(\eta, 0) - Q(\eta, f_\phi^*(\eta))\right), \tag{67}$$

then for any measurable function $f(x)$:

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) \leq 2c \left(\mathcal{E}(f) - \mathcal{E}(f_\rho^V)\right)^{1/s}.$$

Further analysis was made by Bartlett, Jordan and McAuliffe (2003). For example, it was proved in Bartlett, Jordan and McAuliffe (2003, Theorem 6) that for a convex function ϕ, $f_\phi^*(\eta) > 0$ for any $\eta > 0.5$ if and only if ϕ is differentiable at 0 and $\phi'(0) < 0$. Borrowing some ideas from Bartlett, Jordan and McAuliffe (2003), we can derive a simple criterion for the condition (67) with $s = 2$. The existence of $\phi''(0)$ means that the function $\phi'(x)$ is well defined in a neighborhood of 0 and is differentiable at 0. Note that the convexity of ϕ implies $\phi''(0) \geq 0$.

Theorem 34 *Let $\phi : \mathbb{R} \to \mathbb{R}_+$ be a convex function such that $\phi''(0)$ exists. If $\phi'(0) < 0$ and $\phi''(0) > 0$, then (67) holds for $s = 2$. Hence for any measurable function f:*

$$\mathcal{R}(\mathrm{sgn}(f)) - \mathcal{R}(f_c) \leq 2c\sqrt{\mathcal{E}(f) - \mathcal{E}(f_\rho^V)}.$$

Proof By the definition of $\phi''(0)$, there exists some $1/2 \geq c_0 > 0$ such that

$$\left|\frac{\phi'(f) - \phi'(0)}{f} - \phi''(0)\right| \leq \frac{\phi''(0)}{2}, \qquad \forall f \in [-c_0, c_0].$$

This implies

$$\phi'(0) + \phi''(0)f - \frac{\phi''(0)}{2}|f| \leq \phi'(f) \leq \phi'(0) + \phi''(0)f + \frac{\phi''(0)}{2}|f|.$$

If $\eta > 1/2$, then for $0 \leq f \leq c_0$,

$$\frac{\partial Q}{\partial f}(\eta, f) = \eta\phi'(f) - (1-\eta)\phi'(-f) \leq (2\eta - 1)\phi'(0) + \phi''(0)f + \frac{\phi''(0)}{2}f.$$

Thus for $0 \leq f \leq \Delta_\eta := \min\{\frac{-\phi'(0)}{\phi''(0)}(\eta - \frac{1}{2}), c_0\}$, we have

$$\frac{\partial Q}{\partial f}(\eta, f) \leq (2\eta - 1)\phi'(0) + \frac{3}{2}\phi''(0)\frac{-\phi'(0)}{\phi''(0)}(\eta - \frac{1}{2}) \leq \frac{\phi'(0)}{2}(\eta - \frac{1}{2}) < 0.$$

Therefore as a function of the variable f, $Q(\eta, f)$ is strictly decreasing on the interval $[0, \Delta_\eta]$. But $f_\phi^*(\eta) > 0$ is its minimal point, hence

$$Q(\eta,0) - Q(\eta, f_\phi^*(\eta)) \geq Q(\eta,0) - Q(\eta,\Delta_\eta) \geq -\frac{\phi'(0)}{2}(\eta - \frac{1}{2})\Delta_\eta.$$

When $\frac{-\phi'(0)}{\phi''(0)}(\eta - \frac{1}{2}) > c_0$, we have $\Delta_\eta = c_0 \geq 2c_0(\eta - \frac{1}{2})$. Hence

$$Q(\eta,0) - Q(\eta, f_\phi^*(\eta)) \geq \frac{-\phi'(0)}{2} \min\left\{2c_0, \frac{-\phi'(0)}{\phi''(0)}\right\}(\eta - \frac{1}{2})^2.$$

That is, (67) holds with $s = 2$ and

$$c = \max\left\{\frac{\sqrt{2\phi''(0)}}{-\phi'(0)}, \sqrt{\frac{1}{-\phi'(0)c_0}}\right\}.$$

The proof for $\eta < 1/2$ is the same by estimating the upper bound of $\frac{\partial Q}{\partial f}(\eta, f)$ for $f < 0$. ■

Turn to the special loss function $V = V_q$ given in (6) by $\phi(t) = (1-t)_+^q$. Applying Theorem 34, we see that the function ϕ satisfies $\phi'(0) = -q < 0$ and $\phi''(0) = q(q-1) > 0$. This verifies (40) and the constant c can be obtained from the proof of Theorem 34.

References

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.* **68** (1950), 337–404, 1950.

A. R. Barron. Complexity regularization with applications to artificial neural networks. G. Roussa, editor, in *Nonparametric Functional Estimation*, Kluwer, Dortrecht, pages 561–576, 1990.

P. L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, **44**: 525–536, 1998.

P. L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *Annals Statis.*, 2004, to appear.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. Preprint, 2003.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop of Computational Learning Theory*, **5**, Pittsburgh, ACM, pages 144–152, 1992.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, **2**: 499–526, 2002.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, **20**: 273–297, 1995.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, **39**: 1–49, 2001.

F. Cucker and D. X. Zhou. *Learning Theory: An Approximation Theory Viewpoint*. Monograph manuscript in preparation for Cambridge University Press, 2004.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1997.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Adv. Comput. Math.*, **13**: 1–50, 2000.

W. S. Lee, P. Bartlett, and R. Williamson. The importance of convexity in learning with least square loss. *IEEE Transactions on Information Theory*, **44**: 1974–1980, 1998.

Y. Lin. Support vector machines and the Bayes rule in classification. *Data Mining and Knowledge Discovery* **6**: 259–275, 2002.

S. Mendelson. Improving the sample complexity using global data. *IEEE Transactions on Information Theory* **48**: 1977–1991, 2002.

S. Mukherjee, R. Rifkin, and T. Poggio. Regression and classification with regularization. In D. D. Denison, M. H. Hansen, C. C. Holmes, B. Mallick, and B. Yu, eds., *Lecture Notes in Statistics: Nonlinear Estimation and Classification*, Springer-Verlag, New York, pages 107–124, 2002.

P. Niyogi and F. Girosi. On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. *Neural Comp.*, **8**: 819–842, 1996.

M. Pontil. A note on different covering numbers in learning theory. *J. Complexity* **19**: 665–671, 2003.

F. Rosenblatt. *Principles of Neurodynamics*. Spartan Book, New York, 1962.

J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transations on Information Theory* **44**: 1926–1940, 1998.

S. Smale and D. X. Zhou. Estimating the approximation error in learning theory. *Anal. Appl.*, **1**: 17–41, 2003.

S. Smale and D. X. Zhou. Shannon sampling and function reconstruction from point values. *Bull. Amer. Math. Soc.* , **41**: 279–305, 2004.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, **2**: 67–93, 2001.

I. Steinwart. Support vector machines are universally consistent. *J. Complexity*, **18**: 768–791, 2002.

A. Tikhonov and V. Arsenin. *Solutions of Ill-posed Problems*. W. H. Winston, 1977.

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer-Verlag, New York, 1996.

V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

G. Wahba. *Spline models for observational data*. SIAM, 1990.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the Randomized GACV. In Schölkopf, Burges and Smola, eds., *Advances in Kernel Methods - Support Vector Learning*, MIT Press, pages 69–88, 1999.

R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory* **47**: 2516–2532, 2001.

Q. Wu and D. X. Zhou. Analysis of support vector machine classification. Preprint, 2004.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals Statis.*, **32**: 56–85, 2004.

D. X. Zhou. The covering number in learning theory. *J. Complexity*, **18**: 739–767, 2002.

D. X. Zhou. Capacity of reproducing kernel spaces in learning theory. *IEEE Transactions on Information Theory*, **49**: 1743–1752, 2003a.

D. X. Zhou. Density problem and approximation error in learning theory. Preprint, 2003b.

Journal of Machine Learning Research 5 (2004) 1177–1203 Submitted 11/03; Revised 9/04; Published 9/04

Model Averaging for Prediction with Discrete Bayesian Networks

Denver Dash DENVER.H.DASH@INTEL.COM
Intel Research
SC12-303
3600 Juliette Lane
Santa Clara, CA 95054, USA

Gregory F. Cooper GFC@CBMI.UPMC.EDU
Center for Biomedical Informatics
University of Pittsburgh
Pittsburgh, PA 15260, USA

Editor: David Madigan

Abstract

In this paper[1] we consider the problem of performing Bayesian model-averaging over a class of discrete Bayesian network structures consistent with a partial ordering and with bounded in-degree k. We show that for N nodes this class contains in the worst-case at least $\Omega(\binom{N/2}{k}^{N/2})$ distinct network structures, and yet model averaging over these structures can be performed using $O(\binom{N}{k} \cdot N)$ operations. Furthermore we show that there exists a single Bayesian network that defines a joint distribution over the variables that is equivalent to model averaging over these structures. Although constructing this network is computationally prohibitive, we show that it can be approximated by a tractable network, allowing approximate model-averaged probability calculations to be performed in $O(N)$ time. Our result also leads to an exact and linear-time solution to the problem of averaging over the 2^N possible feature sets in a naïve Bayes model, providing an exact Bayesian solution to the troublesome feature-selection problem for naïve Bayes classifiers. We demonstrate the utility of these techniques in the context of supervised classification, showing empirically that model averaging consistently beats other generative Bayesian-network-based models, even when the generating model is not guaranteed to be a member of the class being averaged over. We characterize the performance over several parameters on simulated and real-world data.

Keywords: Bayesian networks, Bayesian model averaging, classification, naïve Bayes classifiers, feature selection

1. Introduction

A probabilistic model M over a set of variables X, is a parameterization of the joint distribution $P(X)$ over X. There are many practical uses for $P(X)$, including the ability to calculate expectations, $E(X)$, of configurations of variables, the ability to calculate the *most likely explanation* of some observed evidence, the ability to *update beliefs* about some variables given some other variables, $P(X_i, X_j \mid X')$, etc. In short virtually any probabilistic quantity of interest involvoing the variables X can be calculated once $P(X)$ is known. Bayesian networks (BNs) (cf., Pearl, 1988) are a popular

1. This is a combined and expanded version of previous conference and workshop papers (Dash and Cooper, 2002, 2003).

class of graphical probabilistic models that allow $P(X)$ to be specified in practice even when $|X|$ is very large, by explicating independence between the variables X.

Many algorithms for learning BNs from data (Verma and Pearl, 1991; Cooper and Herskovits, 1992; Spirtes, Glymour, and Scheines, 1993; Heckerman, Geiger, and Chickering, 1995; Friedman, Geiger, and Goldszmidt, 1997) have been used effectively to learn the structure of a BN model from data, typically by performing a search over structures using the posterior probability, $P(S \mid D)$, of the structure given the data as a measure of quality. While learning a particular BN structure has shown to be useful, it suffers from the fact that a single model makes strong independence assumptions among the variables of interest that may not be true, or may only be approximately true in reality. That is, the process of learning a single network affords no way of capturing the uncertainty in the model structure. The most principled alternative to selecting a particular network structure, is to calculate the full joint posterior $P(X \mid D)$ by averaging over all possible BN structures. Unfortunately, the space of network structures is super-exponential in the number of model variables, and thus an exact method for full model-averaging is likely to be intractable.

One especially common use for learning the joint distribution from data is *supervised classification*. The general supervised classification problem seeks to create a model based on labelled data D, which can be used to classify future vectors of features $F = \{F_1, F_2, \ldots, F_N\}$ into one of various classes of interest. A probabilistic model accomplishes this goal by calculating the posterior probability, $P(C \mid F)$, of the class given the features. One of the simplest probabilistic classifiers for this task is the naïve classifier (cf., Duda and Hart, 1973), which, without inferring any structural information from the database, can still perform surprisingly well at the classification task (Domingos and Pazzani, 1997; Friedman, 1997; Ng and Jordan, 2002). Classification using a single Bayesian network model and no missing feature-vector data can be performed in $O(N)$ time. When the feature vector is incomplete then standard algorithms (e.g., Lauritzen and Spiegelhalter, 1988) for Bayesian network inference can be used for classification. The drawbacks of selecting a single model for classification manifests themselves as over-fitting of the data, leading to poor classification accuracy on future data sets; however, model averaging has been shown to reduce over-fitting and provide better generalization (Madigan and Raftery, 1994).

In this paper we consider the possibility of performing exact and approximate model-averaging (MA) over a particular class of structures rather than over the general space of directed acyclic graphs (DAGs). We show that exact model averaging over the class of BN structures consistent with a partial ordering π and with bounded in-degree k, despite its super-exponential size, can be performed with relatively small time and space restrictions.

There has been other work on making model averaging over Bayesian network structures efficient: Methods for approximate MA classification using both selective pruning (Madigan and Raftery, 1994; Volinsky, 1997) and Monte-Carlo techniques (Madigan and York, 1995) exist and have been shown to improve performance in prediction tasks; however these methods are approximate, and do not have the complexity guarantees that our method possesses. Friedman and Koller (2003) studied the ability to estimate structural features of a network (for example the probability of an arc from X_i to X_j) by performing a MCMC search over orderings of nodes. Their method relied on a decomposition, which they credit to Buntine (1991), that we extend in order to prove our key theoretical result. We discuss this issue in detail in Section 3. Their work differs from ours in two key respects: (1) Their approach does not capture the single-network (and thus the efficiency of calculation) approximation to the MA problem, and (2) They perform model averaging only to calculate the probabilities of structural features, explicitly not for prediction. Other work has been

done (Meila and Jaakkola, 2000; Cerquides and de Màntaras, 2003) on performing exact model averaging for prediction over all tree-structures in $O(N^3)$ time. This research also uses similar assumptions and similar decompositions used by Friedman and Koller (2003) and in this paper. Our calculation is more general in allowing nodes to have more than one parent, but it is less general in that it assumes a partial-ordering of the nodes.

The primary contributions in this paper are as follows: (1) we extend the decomposition of Buntine (1991) to apply to the task of prediction, (2) we show that MA calculations over this class can be reproduced by a single network structure S^* which, if it can be constructed tractably, thereafter allows approximate MA predictions to be performed using standard Bayesian network inference, (3) we show that, for the class of naïve models, calculation of S^* (including parameters) can be performed in linear time in the number of variables, and we demonstrate empirically that model averaging over naïve classifiers improves performance, and (4) we develop a technique to make model averaging practical for arbitrary orderings, and we demonstrate empirically that this technique can result in improved classification (compared to other Bayesian network classifiers) even when no ordering information is known *a priori*.

Aside from the practical issue of achieving accurate predictions, our technique is interesting from an analytical perspective. As an example, recently, Domingos (2000) made an argument based on empirical and theoretical grounds that Bayesian model averaging can actually exacerbate the over-fitting problem in machine learning. Empirically, he shows that rule-learners that approximate pure Bayesian model averaging closer and closer achieve successively higher error rates than a rule-learner that uses the more *ad hoc* technique of bagging. He explains this observation as a consequence of the likelihood's exponential sensitivity to random fluctuations in the data, and surmises that the effect will be significant even for small data sets and will be amplified as the number of models being averaged over increases. Our experiments here present a direct test of this assertion, obtaining results that conflict with the conclusions of Domingos.

In Section 2 we formally frame the problem and state our assumptions and notation, and rederive previous results. In Section 3 we derive the MA solution and show that the MA predictions are approximated by those of a single structure bearing a particular set of parameters. In Section 4 we present the experimental comparisons, and in Section 5 we discuss our conclusions and future directions.

2. Previous Results

In this section we frame the problem, introduce our notation and review relevant previous research, re-deriving the results of Friedman and Koller (2003) and Buntine (1991) and casting them into notation that will allow us to extend them for prediction in Section 3.

2.1 Assumptions and Notation

The general supervised classification problem can be framed as follows: Given a set of features $F = \{F_1, F_2, \ldots, F_N\}$, a set of classes $C = \{C_1, C_2, \ldots C_{N_c}\}$, and a labelled training data set $D = \{D_1, D_2, \ldots, D_{N_D}\}$ generated from some distribution P, construct a model to predict into which class future feature vectors sampled from P are most likely to reside. We use the notation X_i to refer to the nodes when we need to have a uniform notation; we use the convention that $X_i \equiv F_i$ and $X_0 \equiv C$, and we use X to denote the collective set of nodes in the network. A directed graph $G(X)$ is defined as a pair $\langle X, E \rangle$, where E is a set of directed edges $X_i \rightarrow X_j$, such that $X_i, X_j \in X$. If X is a

random variable, we let $Rng(X)$ denote the range of X. We typically use boldface symbols to denote sets and non-boldface type to denote elements of sets, when possible.

We are considering the problem of averaging over the space of *Bayesian network* structures. For a set of variables X, a Bayesian network on X is a graphical model which factorizes the joint distribution $P(X)$ over X. In particular:

Definition 1 (Bayesian network) *Given a set X of N variables, a Bayesian network B on X is a pair $B = \langle S, \theta \rangle$, where $S = \langle X, E \rangle$ is a directed acyclic graph over X, and $\theta = \{\theta_0, \theta_1, \ldots, \theta_N\}$ are the parameters of the network that represent the set of conditional probability distributions for each variable in X given its parents in S.*

We make the following assumptions:

Assumption 1 (Multinomial variables) *Each node X_i represents a discrete random variable with r_i possible states: $Rng(X_i) = \{x_i^1, x_i^2, \ldots, x_i^{r_i}\}$.*

We use Pa_i to denote the parent set of X_i, and we let q_i denote the number of possible joint configurations of parents for node X_i (defining q_i=1 if $Pa_i = \emptyset$), which we enumerate as: $Rng(Pa_i) = \{p_i^1, p_i^2, \ldots, p_i^{q_i}\}$; for example, if X_i has 3 binary parents then $q_i = 8$.

Under the assumption of multinomial variables, a conditional probability distribution θ_i for variable X_i will take the form of a conditional probability table (CPT) with components $\theta_{ijk} = P(X_i = x_i^k \mid Pa_i = p_i^j)$, and for a fixed network structure S, the components θ_{ijk} form the parameters of the Bayesian network model and define the joint distribution over all variables assuming the Markov condition holds. We use the symbol θ_{ij} to denote the entire conditional probability distribution function for the i-th node and the j-th parent configuration, and the symbol θ to denote the collective parameters of the network. In general we use the common (ijk) coordinates notation to identify the *k-th* state and the *j-th* parent configuration of the *i-th* node in the network. We use the shorthand that if Q_{ijk} is some quantity associated with coordinates (ijk), then $Q_{ij} \equiv \sum_k Q_{ijk}$.

Assumption 2 (Complete labelled training data) *The training data set D contains no record $D_l \in D$ such that D_l has a non-observed component.*

We will discuss ways to relax this assumption in Section 5. We let N_{ijk} denote the sufficient statistics of the data set (i.e., the number of times that node X_i achieved state k when parent set Pa_i was in the j-th configuration).

Assumption 3 (Dirichlet priors) *The prior beliefs over parameter values are given by a Dirichlet distribution.*

We let α_{ijk} denote the Dirichlet hyperparameter corresponding to the network parameter θ_{ijk}. For simplicity, we assume α_{ijk} can be calculated in $O(1)$ time and space; this is the case, for example, with two popular metrics, the K2 metric (Cooper and Herskovits, 1992) and the BDeu metric (Heckerman, Geiger, and Chickering, 1995).

Assumption 4 (Parameter independence) *For any given network structure S, each probability distribution θ_{ij} is independent of any other probability distribution $\theta_{i'j'}$:*

$$P(\theta \mid S) = \prod_{i=0}^{N} \prod_{j=1}^{q_i} P(\theta_{ij} \mid S). \tag{1}$$

Finally, we take the assumption that the priors on parameters θ_{ijk} for a node X_i depend only on the local structure. This assumption is known as *parameter modularity* (Heckerman, Geiger, and Chickering, 1995):

Definition 2 (Parameter modularity) *Let X be a set of variables with $X_i \in X$. For any two network structures S_1 and S_2 over X, if $Pa_i|_{S_1} = Pa_i|_{S_2}$ then $P(\theta_{ijk} \mid S_1) = P(\theta_{ijk} \mid S_2)$.*

2.2 Averaging Over Parameters with a Fixed Network Structure

One common goal in machine learning with Bayesian networks is to calculate the probability of a configuration $X = x$ of a set of variables X. This can be used for predicting likely configurations of variables, or it can be queried for any conditional probability of interest over the variables in X (e.g., $P(X_1, X_2 \mid X_3)$) which could be useful for prediction. For a fixed network structure S and a fixed set of network parameters θ, $P(X = x \mid S, \theta)$ can be calculated in $O(N)$ time:

$$P(X = x \mid S, \theta) = \prod_{i=0}^{N} \theta_{iJK}, \tag{2}$$

where all (j,k) coordinates are fixed by the configuration of X to the value $(j,k) = (J,K)$.

When, rather than a fixed set of parameters, a database D is given, it is necessary to average over all possible configurations of the parameters θ:

$$\begin{aligned} P(X = x \mid S, D) &= \int P(X = x \mid S, \theta) \cdot P(\theta \mid S, D) \cdot d\theta \\ &= \int \prod_{i=0}^{N} \theta_{iJK} \cdot P(\theta \mid S, D) \cdot d\theta, \end{aligned}$$

where the second line follows from Equation 2. Given the assumption of parameter independence and Dirichlet priors, this quantity can be written just in terms of sufficient statistics and Dirichlet hyperparameters (Cooper and Herskovits, 1992; Heckerman, Geiger, and Chickering, 1995):

$$P(X = x \mid S, D) = \prod_{i=0}^{N} \frac{\alpha_{iJK} + N_{iJK}}{\alpha_{iJ} + N_{iJ}}, \tag{3}$$

where we have used the notation: $\alpha_{ij} = \sum_k \alpha_{ijk}$. Comparing this result to Equation 2 illustrates the well-known result that a single network with a fixed set of parameters $\tilde{\theta}$ given by

$$\tilde{\theta}_{ijk} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}} \tag{4}$$

will produce predictions equivalent to those obtained by averaging over all parameter configurations. We refer to Equation 4 as the *standard parameterization*.

2.3 Averaging Structural Features with a Fixed Ordering

The decomposition by Buntine used by Friedman and Koller was a dynamic programming solution which calculated, with relative efficiency, for example, the posterior probability $P(X_L \rightarrow X_M \mid D)$

of a particular arc $X_L \rightarrow X_M$ averaged over all in-degree-bounded networks consistent with a fixed ordering. Friedman and Koller then showed how this quantity could be used in a MCMC search for the most likely structural feature, where the search went over orderings instead of DAGs. One might be interested in this quantity, for example, if you were interested in a Bayesian estimate for the structural dependency relations of the system given the data; see Friedman and Koller (2003) for more motivation for why this quantity would be useful. In this section we re-derive the result of Buntine.

The derivation required the ability to calculate efficiently the prior probability $P(S)$ that a given structure S generated the database D. An additional assumption was introduced, labelled *structure modularity* by Friedman and Koller:

Assumption 5 (Structure modularity) *The prior of a structure S, $P(S)$, can be factored according to the network*

$$P(S) \propto \prod_{i=0}^{N} p_s(X_i, Pa_i), \tag{5}$$

where $p_s(X_i, Pa_i)$ is some function that depends only on the local structure (X_i and Pa_i).

Any metric that assesses the structure prior $P(S)$ based on a difference in arcs between S and some prior network structure S' (as suggested by Heckerman, Geiger, and Chickering (1995)) will satisfy this condition. Also the uniform distribution will obviously satisfy this assumption, and requires $O(1)$ time to assess. Obviously, we restrict ourselves to structures that give probability zero to a non-acyclic DAG.

The posterior probability $P(X_L \rightarrow X_M \mid D)$ can be written as

$$\begin{aligned} &P(X_L \rightarrow X_M \mid D) = \\ &\quad c \sum_S \delta_K(X_L \rightarrow X_M \in S) \cdot P(D \mid S) \cdot P(S), \end{aligned} \tag{6}$$

where $c = 1/P(D)$ is a constant that depends only on the database, and $\delta_K(Z)$ is the Kronecker delta function:

$$\delta_K(Z) = \begin{cases} 1 & \text{if } Z = \textit{true} \\ 0 & \text{otherwise.} \end{cases}$$

The summation in Equation 6 includes a super-exponential number of network structures, and therefore appears to be intractable. Buntine handled this problem by imposing a total ordering on the nodes and restricting the maximum number of parents, k, that each node can have. Generalizing his results to a partial ordering π instead of a total ordering is straightforward, and we do that here. For a given partial ordering π and a particular node X_i, it is required to enumerate all of X_i's possible parent sets up to a maximum size k. To this end, we will typically use the superscript ν to index the different parent sets. For example, four nodes partitioned as $\pi = \langle \{X_1, X_3\}, \{X_2, X_4\} \rangle$ and a maximum in-degree $k = 2$ would yield the following enumeration of parent sets for X_2: $\{Pa_2^0 = \emptyset, Pa_2^1 = \{X_1\}, Pa_2^2 = \{X_3\}, Pa_2^3 = \{X_1, X_3\}\}$.

The class of models consistent with π with bounded in-degree of k we denote as $\mathcal{L}_k^\pi$:

Definition 3 ($\mathcal{L}_k^\pi$) *For a given integer $k \leq N$ and a given partial ordering π of X, a DAG $G = \langle X, E \rangle \in \mathcal{L}_k^\pi$ iff arcs are directed from higher to lower levels and no variable has more than k parents: $X_i \rightarrow X_j \in E \Rightarrow \pi(X_i) > \pi(X_j)$, and $X_i \in X \Rightarrow |Pa_i| \leq k$.*

Note that if $D_1, D_2 \in \mathcal{L}_k^\pi$ and $D_1 \neq D_2$ then D_1 is statistically distinguishable from D_2 because it will include a different set of adjacencies (Verma and Pearl, 1991); so when averaging over the class $\mathcal{L}_k^\pi$, we are never averaging over equivalent DAGs.

The number of structures in $\mathcal{L}_k^\pi$ is still exponentially large in the worst-case; each node at level l can choose up to k parents from among all the nodes in levels $l' > l$. For $k \leq N/2$, $\mathcal{L}_k^\pi$ in the worst-case includes at least $\Omega\left[\binom{N/2}{k}^{N/2}\right]$ network structures. This result corresponds to the case where π consists of two levels, each with $N/2$ nodes; each of the $N/2$ nodes in the bottom level can therefore choose up to $\binom{N/2}{k}$ possible parents.[2]

Given the assumptions of this paper, $P(D \mid S)$, can be written just in terms of hyperparameters and sufficient statistics (Cooper and Herskovits, 1992; Heckerman, Geiger, and Chickering, 1995):

$$P(D \mid S) = \prod_{i=0}^{N} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}. \tag{7}$$

Given structure modularity (Assumption 5) and Equation 7, Equation 6 can be written as

$$P(X_L \rightarrow X_M \mid D) = c \sum_S \prod_{i=0}^{N} \rho_{iLM}^S. \tag{8}$$

The ρ_{iLM}^S functions are given by

$$\rho_{iLM}^S = \delta_K[M \neq i \vee X_L \in Pa_i] \cdot p_s(X_i, Pa_i) \cdot \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}, \tag{9}$$

and can be calculated using information that depends only on nodes X_i and Pa_i.

Even restricting structures to those in a particular $\mathcal{L}_k^\pi$ class, as already mentioned, the summation in Equation 8 still contains an exponential number of structures in the worst-case. However, the following theorem (similar to one from Buntine (1991)) shows that $P(X_L \rightarrow X_M \mid D, \mathcal{L}_k^\pi)$ can be calculated with relative efficiency:

Theorem 1 *Assuming $S \in \mathcal{L}_k^\pi$, equation 8 can be written as $P(X_L \rightarrow X_M \mid D, \mathcal{L}_k^\pi) = c \prod_{i=0}^{N} \sum_{\nu=0}^{\mu_i} \rho_{iLM}^\nu$, where ρ_{iLM}^ν denotes ρ_{iLM}^S for the νth parent set Pa_i^ν of X_i and the summation goes over all parent sets available to node X_i under the restrictions of $\mathcal{L}_k^\pi$.*

Proof: For notational simplicity, we will drop the LM subscripts on the ρ functions: $\rho_i^\nu \equiv \rho_{iLM}^\nu$. Expanding the sum in Equation 8 yields

2. We are not asserting that this two-level ordering is the absolute worst-case, only that the worst-case must have at least this many network structures.

$$P(X_L \to X_M \mid D, \mathcal{L}_k^\pi) \propto \left.\begin{array}{rl} & \rho_0^0 \cdot \rho_1^0 \cdot \ldots \cdot \rho_N^0 \\ + & \rho_0^1 \cdot \rho_1^0 \cdot \ldots \cdot \rho_N^0 \\ \vdots & \vdots \\ + & \rho_0^{\mu_0} \cdot \rho_1^0 \cdot \ldots \cdot \rho_N^0 \\ + & \rho_0^0 \cdot \rho_1^1 \cdot \ldots \cdot \rho_N^0 \\ + & \rho_0^1 \cdot \rho_1^1 \cdot \ldots \cdot \rho_N^0 \\ \vdots & \vdots \\ + & \rho_0^{\mu_0} \cdot \rho_1^1 \cdot \ldots \cdot \rho_N^0 \\ \vdots & \vdots \\ + & \rho_0^0 \cdot \rho_1^0 \cdot \ldots \cdot \rho_N^1 \\ + & \rho_0^1 \cdot \rho_1^0 \cdot \ldots \cdot \rho_N^1 \\ \vdots & \vdots \\ + & \rho_0^{\mu_0} \cdot \rho_1^0 \cdot \ldots \cdot \rho_N^1 \\ \vdots & \vdots \\ + & \rho_0^{\mu_0} \cdot \rho_1^{\mu_1} \cdot \ldots \cdot \rho_N^{\mu_N} \end{array}\right\} \Omega\left[\binom{N/2}{k}^{N/2}\right] \text{ terms, worst-case.}$$

We define the symbol Σ_m to denote the structure sum of the product up to and including the m-th node:

$$\begin{array}{rcl} \Sigma_m & \equiv & \rho_0^0 \cdot \rho_1^0 \cdot \ldots \cdot \rho_m^0 \\ & + & \rho_0^1 \cdot \rho_1^0 \cdot \ldots \cdot \rho_m^0 \\ & \vdots & \vdots \\ & + & \rho_0^{\mu_0} \cdot \rho_1^{\mu_1} \cdot \ldots \cdot \rho_m^{\mu_m}. \end{array}$$

Using this notation, the following recursion relation can be derived:

$$\Sigma_i = \Sigma_{i-1} \cdot \sum_{\nu=1}^{\mu_i} \rho_i^\nu, \qquad \Sigma_{-1} = 1.$$

Finally, expanding the recurrence relation yields the expression for $P(X_L \to X_M \mid D, \mathcal{L}_k^\pi)$:

$$P(X_L \to X_M \mid D, \mathcal{L}_k^\pi) = c \prod_{i=0}^{N} \sum_{\nu=1}^{\mu_i} \rho_i^\nu. \tag{10}$$

□

Thus, a summation of $\Omega\left[\binom{N/2}{k}^{N/2}\right]$ terms can be performed in $O\left[\binom{N}{k} \cdot N\right]$ operations.

3. Model Averaging for Prediction

In this section we show how to extend the results of Section 2 to efficiently calculate the quantity $P(X = x \mid D, \mathcal{L}_k^\pi)$ averaged over the class $\mathcal{L}_k^\pi$.

This quantity can be written as

$$P(X = x \mid D, \mathcal{L}_k^\pi) = \frac{1}{P(D)} \sum_{S \in \mathcal{L}_k^\pi} \prod_{i=0}^{N} \tilde{\theta}_{iJK} \cdot P(D \mid S) \cdot P(S), \tag{11}$$

where $\tilde{\theta}_{iJK}$ are the standard parameters given in Equation 4. Given structure modularity and Equation 7, Equation 11 can be written in a form very similar to Equation 8:

$$P(X = x \mid D, \mathcal{L}_k^\pi) = c \sum_{S \in \mathcal{L}_k^\pi} \prod_{i=0}^{N} \tilde{\rho}_{iJ_{\mathbf{x}}^S K_{\mathbf{x}}}, \tag{12}$$

where here the $\tilde{\rho}_{iJ_{\mathbf{x}}^S K_{\mathbf{x}}}$ functions are given by

$$\tilde{\rho}_{iJ_{\mathbf{x}}^S K_{\mathbf{x}}} = \tilde{\theta}_{iJ_{\mathbf{x}}^S K_{\mathbf{x}}} \cdot \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})} \cdot p_s(X_i, Pa_i), \tag{13}$$

and c is a constant (not dependent on S) equal to $1/P(D)$. We subscript the indices J and K from Equation 2 with an x to indicate that they are fixed by a particular configuration of X, and J is indexed by S to emphasize that the value of the parent configuration index depends on the number of parents and therefore the structure S of the network. Although this notation may seem cumbersome, we believe it clarifies the analysis later.

As in Section 2.3, the worst-case number of terms in the summation of Equation 12 is exponential in the number of features N. The $\tilde{\rho}_{iJ_{\mathbf{x}}^S K_{\mathbf{x}}}$ functions again can be calculated using only information local to node X_i and Pa_i. Following a derivation identical to that for averaging $P(X_L \rightarrow X_M \mid D, \mathcal{L}_k^\pi)$ in Section 2.3, yields the following :

$$P(X = x \mid D, \mathcal{L}_k^\pi) = c \prod_{i=0}^{N} \sum_{\nu=1}^{\mu_i} \tilde{\rho}_{iJ_{\mathbf{x}}^\nu K_{\mathbf{x}}}^{\nu}. \tag{14}$$

Here the S index has been replaced with a ν indicating which parent set for node X_i is being considered. The following theorem shows that this summation can be performed in $O(\binom{N}{k} \cdot N \cdot k \cdot N_D \cdot R)$ time and $O(k \cdot N_D)$ space.

Theorem 2 *For N variables with a maximum number of states per variable given by R and a database of N_D records, Equation 14 can be calculated in $O(\binom{N}{k} \cdot N \cdot k \cdot N_D \cdot R)$ time and using $O(k \cdot N_D)$ space.*

Proof: The right-hand-side of Equation 14 includes N products and $\binom{N}{k}$ sums. Each ρ_{iLM}^{ν} term can be calculated in $O(k \cdot N_D \cdot R)$ time and $O(k \cdot N_D)$ space. This result follows because all sufficient statistics for a given node X_i can be stored in a tree of depth $O(k)$ and width $O(N_D)$, with the leaves of the tree holding the sufficient statistic for the given configuration of X_i and $Pa(X_i)$. To fill the tree requires N_D passes of the tree, thus taking $O(k \cdot N_D)$ time. Once the tree is constructed, it can be queried for any statistic in $O(k)$ time.

The number of possible parent configurations present in the data are bound by the number of records N_D; thus the number of (i, j) configurations for which $N_{ij} \neq 0$ is $O(N_D)$. Furthermore, all terms in the product of Equation 14 for which $N_{ij} = 0$ will equal 1 so will not contribute to the product. Thus, the calculation of products in Equation 13 require $O(k \cdot N_D \cdot R)$ time.

Putting all the steps together yields the claim of the theorem. □

Although this result is relatively efficient, for classification purposes it may still be too complex of a calculation. The functional form of the above solution allows us to prove that exact model averaging over $\mathcal{L}_k^\pi$ can actually be performed with a single Bayesian network structure:

Theorem 3 *There exists a single Bayesian network model $M^* = \langle S^*, \theta^* \rangle$ that will define a joint distribution $P(X = x \mid S^*, \theta^*)$ that is equivalent to that produced by model averaging over all models in $\mathcal{L}_k^\pi$.*

Proof: Let S^* be defined so that each node X_i has the parent set $Pa_i^* = \bigcup_{\nu=1}^{\mu_i} Pa_i^\nu$, and let θ^* be defined by

$$\theta_{ijk}^* = c^{1/N} \sum_{\nu=1}^{\mu_i} \tilde{\rho}_{iJ_j^\nu k}^\nu, \tag{15}$$

where the x subscript for $J_{\mathbf{x}}^\nu$ has now been replaced with a j and $K_{\mathbf{x}}$ has been replaced with a k subscript, because we are considering a particular coordinate (ijk). It can be seen by direct comparison that substituting θ_{ijk}^* into Equation 2 will yield Equation 14. □

If we define functions $f(X_i, Pa_i^\nu \mid D)$ such that $f(X_i, Pa_i^\nu \mid D) = \tilde{\rho}_{iJ_j^\nu k}^\nu / \tilde{\theta}_{iJ_j^\nu k}^\nu$, then Equation 15 can be written as

$$\theta_{ijk}^* = c^{1/N} \sum_{\nu=1}^{\mu_i} \tilde{\theta}_{iJ_j^\nu k}^\nu \cdot f(X_i, Pa_i^\nu \mid D). \tag{16}$$

The functions $f(X_i, Pa_i^\nu \mid D)$ do not depend on the indices J_j^ν and k, and they can be interpreted as the local posterior probability that the parent set of X_i is in fact Pa_i^ν. Equation 16 thus provides the interpretation that M^* represents a structure-based smoothing where each standard parameter $\tilde{\theta}_{ijk}^\nu$ is weighted based on the likelihood that Pa_i^ν is the parent set of X_i. Since θ_{ijk}^* is interpretable as a probability, the constant $c^{1/N}$ serves as a normalization constant and need not be calculated directly.

There are some numerical complexities with calculating the parameters using Equation 16. One must essentially calculate quantities of the form

$$\theta_i^* = c^{1/N} \sum_\nu \tilde{\theta}_i^\nu \cdot \exp lf_i^\nu, \tag{17}$$

where $lf_i^\nu = \log f_i^\nu$ is a negative number with large absolute value. Exponentiating this value will usually be truncated to zero using floating-point arithmetic. In reality however, the normalization constant $c^{1/N}$ is also very small and in fact makes θ_i^* nonzero in many cases. To get around this problem, we use a known trick of shifting the exponentials so the largest term is equal to 1. The net result of this is to change the normalization constant, which is never calculated explicitly, i.e.

$$c^{1/N} \sum_\nu \tilde{\theta}_i^\nu \cdot \exp lf_i^\nu = \frac{c^{1/N}}{\exp(-lf_{max})} \sum_\nu \tilde{\theta}_i^\nu \cdot \exp(lf_i^\nu - lf_{max}), \tag{18}$$

Theorem 3 implies that, rather than performing the $O(\binom{N}{k} \cdot N)$ summation in Equation 14 for each case to be classified, in principle we need only construct a single model M^* and use standard Bayesian network inference for each case. In the case of a completely instantiated feature vector, this inference can be completed in $O(N)$ time; otherwise BN inference can be used. Unfortunately for almost all realistic partial orderings M^* will be a highly dense network and memory requirements will be prohibitive. Furthermore, the network structure S^* would be non-interpretable by a human. The intractability of M^* is a central problem with applying this method in practice, and we devote most of the remainder of this paper to presenting ways to remedy this problem.

3.1 Model Averaging over Naïve structures

One popular class of models that fits into the $\mathcal{L}_k^\pi$ schema is the class of naïve (simple) Bayes models. The naïve classifier is a probabilistic model that accomplishes classification by making the assumption that any feature $F_i \in F$ is conditionally independent of any other feature $F_j \in F$ given the value of the class variable C. The naïve model can be represented by the Bayesian network shown in Figure 1.

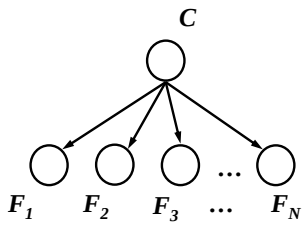


Figure 1: A naïve network: C is the class node which can take on one value for each possible class, and the F_i denote features of interest.

Naïve classifiers have several desirable features: First, they are simple to construct, requiring very little domain background knowledge, as opposed to general Bayesian networks which can require numerous intensive sessions with experts, or a large real-world database to learn the dependence structure between features. Second, naïve networks can be built with very constrained space and time complexity: constructing a network requires the estimation of a set of $O(N \cdot R \cdot N_c)$ parameters, where R is the maximum number of feature states and N_c is the number of class states. Each of which can be estimated from data in time $O(N_D)$, where N_D is the number of records in the database. Inference with naïve networks is also efficient; classification of a new feature vector F' can be performed in time $O(|F'|)$, even if F' is an incomplete instantiation of features.

Despite their simplicity, these classifiers have been shown to perform surprisingly well in practice. Domingos and Pazzani (1997) have shown that naïve classifiers can be optimal (in terms of classifcation accuracy) even when the underlying distribution does not satisfy the naïve assumptions. Friedman (1997) argues that the low variance of the naïve classifier can mitigate the bias, resulting in overall accurate predictions. Finally, Ng and Jordan (2002) show both theoretically and empirically that the naïve classifier converges quickly to its asymptotic error-level. These studies explain why the naïve model has continued to compare favorably to state-of-the-art classification algorithms.

The construction of a naïve classifier given a set F of potential attributes requires only two general steps: (1) Select the subset of features $F' \subseteq F$ judged to be relevant to classification, and (2) Calculate the set $\tilde{\theta}$ of parameters using Equation 4. The feature selection problem (1) is a difficult and central problem in machine learning in general. In terms of naïve classifiers, the selection of the appropriate subset F' has been shown to be both important to classification and non-trivial to perform in practice (Langley and Sage, 1994; Kohavi and John, 1997; Friedman, Geiger, and Goldszmidt, 1997). Obviously eliminating features that do not bear on the classification is important, but also important is the ability to minimize redundant features.

Our method allows us to take a strict Bayesian approach to feature selection; rather than finding a single "good" set F', we can efficiently address the problem of model averaging predictions over

all 2^N possible feature-set structures. An enumeration of these different structures is illustrated in Figure 2.

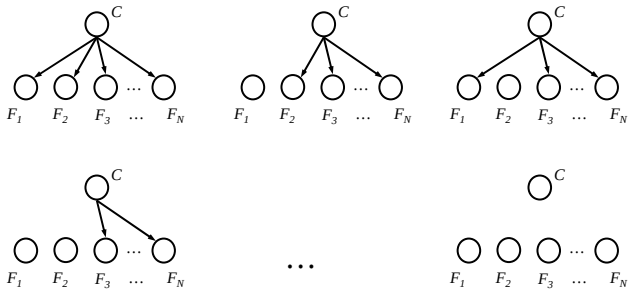


Figure 2: Enumerating all the 2^N possible naïve Bayes net structures.

The summary model M^*, defined in Theorem 3, for the naïve class is especially simple, itself being a naïve network over *all* features. Equation 16 for a naïve Bayes net reduces to

$$\theta^*_{ijk} \propto \tilde{\theta}^{\emptyset}_{ijk} \cdot f_i(\emptyset \mid D) + \tilde{\theta}^{C}_{ijk} \cdot f_i(\{C\} \mid D), \tag{19}$$

where $f_i(\emptyset \mid D)$ and $f_i(\{C\} \mid D)$ are proportional to the local posterior probability of $Pa_i = \emptyset$ and $Pa_i = \{C\}$, respectively. The sufficient statistics and Dirichlet priors required for this calculation are the same as those needed for calculating the parameters of (a) a single network with no arcs present, and (b) a single naïve network with all arcs present. This reparameterization requires order $O(N \cdot N_D)$ time and space requirements, which are the same that are needed to calculate the standard parameters of a single naïve network over all N features. We call a naive structure so parameterized a *Naïve model averaging* (NMA) classifier. In Section 4 we present empirical results showing that this reparameterization, over a wide range of experimental parameters, almost always produced better classification results than a standard naïve model.

3.2 Approximate Model Averaging

As noted in Section 3, a serious practical difficulty with constructing M^* according to Theorem 3 when no ordering is known, is that it requires in the worst case the construction of a completely-connected Bayesian network and inference can thus be intractable for all but small N. An obvious pruning strategy, however, is to truncate the sum in Equation 15 to include no more than n parents. Here we present one possible method for selecting the n most important parents for each node.

If we reorder the possible parent sets for node X_i as $O_P \equiv \{Pa_i^1, \ldots, Pa_i^{\mu_i}\}$ such that $f(X_i, Pa_i^\nu \mid D) > f(X_i, Pa_i^\lambda \mid D)$ if and only if $\nu < \lambda$, then an approximation for Pa_i^* can be constructed by the following procedure:

Procedure 1 (Approximate P_i^* construction)
Given: n and O_P.

1. *Let* $Pa_i^* = \emptyset$
2. *For* $\nu = 1$ *to* μ_i,
   if $|Pa_i^* \cup Pa_i^\nu| \leq n$, *let* $Pa_i^* = Pa_i^* \cup Pa_i^\nu$, *else continue.*

We denote the class of structures being averaged over using this procedure as $\mathcal{L}^{\pi}_{kn}(D)$, and we call the method *Approximate Model Averaging* (AMA). Obviously $\mathcal{L}^{\pi}_{kN}(D) = \mathcal{L}^{\pi}_{k}$. Furthermore, we empirically show in Section 4 that the loss in ROC area, ε, due to this approximation for $n \geq 10$ lies around $-0.6\% \leq \varepsilon \leq 0.6\%$ with 99% confidence for $N \leq 100$ and for a wide range of other parameters. We also show empirically that classifications are not typically sensitive to the value of n, so often n can be made relatively small without degrading classification results.

4. Experimental Tests

In this section we describe several experimental investigations that were designed to test the performance of NMA and AMA on arbitrary distributions. We first generate synthetic data to allow us to more extensively vary parameters, then we perform tests on several real-world machine learning data sets. All experiments were implemented in C++ using code that was based on the *Structural Modelling, Inference and Learning Engine* (SMILE) (Druzdzel, 1999), a library for constructing probabilistic decision support models.[3] Experiments were run on an 1.6 GHz Pentium PC with 1 GB of RAM running Windows XP.

4.1 Experimental Setup

There are at least five parameters for which we sought to characterize the performance of classifier predictions: the number of nodes N, the approximation limit n on the size of the maximum in-degree in the summary network, the maximum in-degree ("density") K of the *generating* network, the maximum in-degree k ($k \leq n$) allowed in models in $\mathcal{L}^{\pi}_{k}$, and the number of records N_D. It is beyond the scope of this paper to present a comprehensive comparison over this full five-dimensional space; however, here we sample their settings to provide insight into the dependence of the results on these parameters.

In our experiments, four classifiers were compared: AMA using a fixed partial ordering, a NMA classifier, a single naïve network (SNN) with the standard parameterization (Domingos and Pazzani, 1997), and a non-restricted two-stage greedy thick-thin (GTT) model selection over the space of DAGs, which is described below.

The algorithm used to generate the GTT model, which assumes no ordering on the nodes, is as follows:

Procedure 2 (Greedy thick-thin search)
Given: a network S with no arcs.
Do:

1. *Repeatedly add the arc whose addition maximally increases the marginal likelihood $P(D \mid S)$ without creating a cycle until no increase is possible.*

2. *Repeatedly delete the arc whose deletion maximally increases $P(D \mid S)$ until no increase is possible.*

The inner-loop of each test performed the same procedure: Given the six parameters $\{N, N_D, N_{test}, K, n, k\}$, we did the following:

3. SMILE can be downloaded from http://www.sis.pitt.edu/~genie; however the learning functionality required for our experiments is not yet available for public release.

Procedure 3 (Basic testing loop)
Given: N, N_D, N_{test}, K, n, k.
Do:

1. *Generate a random Bayesian network* $B = G(N,K)$.
2. *Sample* N_D *training records and* N_{test} *test records from the distribution defined by B.*
3. *Train two classifiers to be compared,* M_1 *(typically the AMA classifier) and* M_2 *(the classifier to be compared), on the training records.*
4. *Test* M_1 *and* M_2 *on the test data, measuring the ROC areas* R_1 *and* R_2*, respectively, of each.*
5. *Calculate the quantity* $\delta = \frac{R_1 - R_2}{1 - R_2}$.

The *ROC Area* of a classifier (cf., Egan, 1975), is the area of the curve showing the true-positives of the classifier versus the false-positives as the sensitivity of the classifier is swept out from 0 to 1. It has been used with increasing frequency in machine learning because it provides an objective evaluation of a classifier without requiring the specification of a particular utility function (e.g. zero-one loss).

The performance metric δ indicates what percentage of M_2's missing ROC area $(1 - R_2)$ is covered by M_1: If M_1 is perfect then δ will be 1, if M_1 is equivalent to M_2 then δ will be 0, and if M_1 is worse than M_2 ther

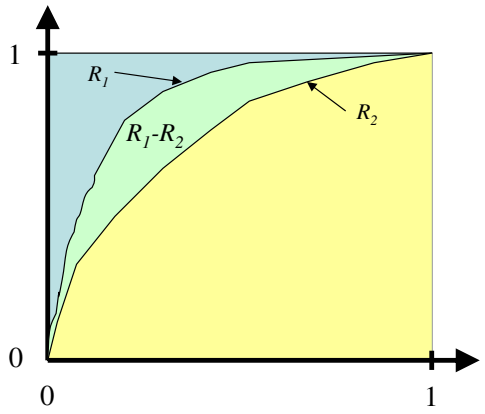


Figure 3: The performance metric δ used in our experiments measures the fraction of ROC area captured by our classifier (with ROC area R_1) versus some other classifier (with ROC area R_2).

parameters, this procedure was repeated N_{trials} times and δ was averaged over these trials.

For some experiments it was necessary to generate networks randomly. We employed a lazy data generation procedure whereby node conditional probability distributions were generated only when they were required by the sampling, a technique which allows generation of data for arbitrarily dense networks. The algorithm for selecting network structures at random is as follows:

Procedure 4 (random structure generation)
Given: a set of nodes V; number of records, N_D; and maximum in-degree, K.
Do:

1. *Construct a total ordering $o(V)$ over the variables.*

2. *For each node V_i do:*

   (a) *Choose a number of parents, n_p, uniformly at random from $\{1,\ldots,K\}$.*

   (b) *Select, uniformly at random, $\min(n_p, o(V_i))$ parents P such that $o(P) < o(V_i)$.*

The choice of the proper set of noninformative structure priors is non-trivial, and in these experiments we do not attempt to address the subtle complexities inherent in this process. In all cases we assume a uniform prior over non-forbidden structures and thus allow $p_s(X_i, Pa_i) = 1/\mu_i$ for all i. These priors will put overwhelming mass on networks with a "medium" number of arcs because there exist many more of these DAGs. We also adopted the K2 parameter prior which sets $\alpha_{ijk} = 1$ for all (i,j,k). This criterion has the property of weighting all local distributions of parameters uniformly, and has been shown empirically to be an effective non-informative prior (Cooper and Herskovits, 1992; Heckerman et al., 1995). All variables in our synthetic tests were binary, $N_c = R = 2$, and $N_{test} = 1000$, and we typically chose an exponentially increasing sequence of values for N to test the benefits of the algorithms on a wide-range of scenarios. In many experiments we sampled the in-degree of generating graphs K uniformly from the set $\{1,2,\ldots,N\}$; we use the notation $K \hookleftarrow \{1,\ldots,N\}$ to denote this procedure.

In all experiments, π for AMA was chosen to be a fixed total ordering of the variables. At least three heuristics were used to generate π: (1) generate a random ordering, (2) generate two opposite random orderings and average predictions of each, and (3) use a topological sort of the graph obtained by GTT. In preliminary experiments, these methods produced comparable results, but (2) and (3) performed a few percent better. In all results presented below, method (3) was used to generate π.

All abbreviations and symbols are summarized in Table 1 as a reference for the reader.

Symbol	*Description*
N	Number of nodes
N_D	Number of training records
N_{test}	Number of testing records
N_{trials}	Number of times Procedure 3 was repeated
K	Maximum in-degree of generating graphs
k	Maximum in-degree in $\mathcal{L}_k^\pi$
n	Maximum in-degree in summary MA network (Section 3.2)
SNN	Single naïve network (with feature selection)
NMA	Naïve model averaging
GTT	Greedy thick-thin
AMA	Approximate model averaging

Table 1: Table of symbols relevant to experiments.

4.2 Experimental Results

In this section we present a range of experimental tests we performed using data generated from random graphs, data generated from the standard benchmark ALARM network, and data from real-world data sets from the UCI database.

4.2.1 NAÏVE MODEL AVERAGING VERSUS A SINGLE NAIVE NETWORK

It has been shown (Domingos and Pazzani, 1997) that the naïve classifier with the standard parameterization can be optimal under zero-one loss even when the underlying independence assumptions are incorrect. That, together with the fact that calculating parameters for this model can be done with a single pass through the data, make it a useful and widely used model for classification, and it would be interesting if we find a model that has the same ease of calculation but performed significantly better. There are, of course, many other classifiers that we could compare to, in particular, optimizing the conditional likelihood under the naïve network assumptions corresponds to a logistic regression model which has also shown to do well for classification. We defer comparisons of our method to logistic regression and other models for future work.

As Section 3.1 shows, using NMA it is possible to model average over all features sets for a naïve model simply by reparametrizing a single naïve network according to Equation 19. Because of the simplicity of this technique (an existing naïve classifier could trivially be replaced with a model-averaging version just by a change in parameters), we performed an extensive set of comparisons of NMA versus a SNN using synthetic data generated from randomly-constructed (not necessarily naïve) Bayesian networks.

Feature selection for the SNN was performed by successively adding the arc that maximized the posterior probability of the network structure S until no arc resulted in an increase. Although this is a greedy strategy, for SNN under the assumptions taken in this paper, it results in a structure that globally maximizes the posterior probability given the data. This can be seen because, given structure modularity, the marginal likelihood for a given arc is independent of all other arcs in the network.

We tested the relative performance of SNN versus a single naïve model (SNN_{all}) over *all* features (i.e., without any feature selection) by performing Procedure 3 with N varied over the set $\{10, 20, 40, 80, 100, 320\}$, with N_D varied over the set $\{100, 200, 400, 800, 1600, 3200, 6400\}$ and with $K \hookleftarrow [1,\ldots,N]$. We found that for all configurations of experimental parameters, SNN dominated SNN_{all}, with average $\delta = 29\% \pm 1\%$. Because of this, we do not include SNN_{all} in any of our comparisons to AMA, NMA or GTT.

In the evaluation of NMA performance over SNN performance, we simultaneously varied N from the set $\{10, 20, 40, 80, 160\}$, N_D from the set $\{50, 100, 200, 400, 800, 1600, 3200, 6400\}$, and K from the set $\{5, 10, 20, 40, 80, 160\}$ (obviously however $K \leq N$). We used $N_{trials} = 1000$ in order to establish clear statistical significance. The results are shown in Table 2. The remaining area covered by NMA ranged from a fraction of one percent to 17%, but in all save one of the 160 configurations measured, the improvement of NMA was significant at the 99% level, the general trend being that NMA performed better for smaller values of N and N_D. The lower and upper quartiles (i.e., the 25% and 75% quantiles: the values of δ that confine the middle 50% of the values of δ that we observed in our tests) show the true spread of the data, i.e., because of the large number of trials, the confidence intervals in the mean are not indicative of the widths of the distributions.

4.2.2 MODEL AVERAGING OVER $\mathcal{L}^{\pi}_{kn}(D)$ VERSUS $\mathcal{L}^{\pi}_{k}$

Since, as we have already mentioned, averaging over the full class $\mathcal{L}^{\pi}_{k}$ requires considerable calculation using Equation 14, we must resort in general to using the single summary network and instead only averaging over the class $\mathcal{L}^{\pi}_{kn}(D)$. We were interested in testing what price we pay in terms of classification for reducing the size of the space of models. The first set of experiments we performed tested the degree of error incurred by model averaging over $\mathcal{L}^{\pi}_{kn}(D)$ instead of the full class $\mathcal{L}^{\pi}_{k}$.

$N_D \rightarrow$		50			100			200			400		
N	K	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u
10	5	17.2 ± 1.6	8.4	24.0	17.0 ± 1.4	9.5	22.8	16.5 ± 1.4	8.7	21.7	14.0 ± 1.2	7.3	19.4
10	10	17.1 ± 1.6	7.7	23.6	16.9 ± 1.4	9.2	23.9	16.2 ± 1.4	7.9	23.5	15.5 ± 1.3	7.8	21.2
20	5	8.7 ± 1.1	3.1	12.1	8.7 ± 1.0	3.7	12.2	8.4 ± 0.9	4.1	11.3	9.0 ± 1.0	3.9	13.3
20	10	8.1 ± 1.0	3.2	12.4	10.0 ± 1.3	3.5	13.5	9.6 ± 1.0	4.7	12.5	8.9 ± 1.1	3.3	11.9
20	20	8.5 ± 1.0	3.7	11.9	9.6 ± 1.1	3.8	13.9	10.1 ± 1.1	4.4	13.7	9.8 ± 1.1	4.2	13.4
40	5	3.8 ± 0.8	0.3	7.3	3.5 ± 0.6	0.8	6.2	4.5 ± 0.6	1.7	6.9	4.6 ± 0.6	1.5	5.7
40	10	3.0 ± 0.6	0.2	5.5	4.2 ± 0.6	1.3	6.6	3.8 ± 0.6	0.8	5.7	4.6 ± 0.7	1.5	6.2
40	20	3.0 ± 0.6	0.2	4.8	4.6 ± 0.7	1.5	6.9	4.4 ± 0.9	1.1	6.7	6.1 ± 1.0	2.1	7.9
40	40	2.2 ± 0.5	-0.2	4.2	3.5 ± 0.7	0.6	5.7	4.6 ± 0.7	1.4	6.8	5.9 ± 0.8	2.0	8.1
80	5	2.8 ± 0.7	-0.4	4.9	2.7 ± 0.6	-0.2	5.0	2.4 ± 0.5	0.1	4.3	2.1 ± 0.4	0.3	3.8
80	10	1.6 ± 0.5	-0.6	2.9	1.5 ± 0.5	-0.8	3.7	2.1 ± 0.4	0.0	3.7	2.6 ± 0.4	0.6	4.4
80	20	1.2 ± 0.5	-0.7	2.6	1.3 ± 0.5	-0.7	2.9	1.9 ± 0.4	-0.1	3.6	2.4 ± 0.5	0.2	4.0
80	40	0.7 ± 0.5	-1.1	2.1	1.0 ± 0.4	-1.1	2.8	1.4 ± 0.5	-0.5	3.1	2.3 ± 0.5	0.1	3.9
80	80	0.7 ± 0.5	-1.0	1.9	1.0 ± 0.4	-0.8	2.6	1.8 ± 0.4	-0.1	3.1	2.4 ± 0.5	0.1	4.4
160	5	2.0 ± 0.6	-0.9	3.6	2.2 ± 0.6	-0.9	4.5	2.2 ± 0.5	-0.1	4.1	1.6 ± 0.4	-0.3	3.2
160	10	1.2 ± 0.5	-1.0	2.8	1.8 ± 0.5	-0.5	3.8	1.6 ± 0.4	-0.6	3.8	1.2 ± 0.4	-0.6	2.9
160	20	1.1 ± 0.5	-0.5	2.2	0.9 ± 0.4	-1.0	2.6	1.0 ± 0.4	-0.7	2.9	0.9 ± 0.4	-1.0	2.6
160	40	0.6 ± 0.4	-1.0	1.7	0.5 ± 0.4	-1.3	2.0	0.5 ± 0.4	-1.0	1.9	0.8 ± 0.4	-1.1	2.4
160	80	0.6 ± 0.4	-1.1	1.6	0.5 ± 0.4	-1.3	2.0	0.5 ± 0.3	-1.2	1.9	0.7 ± 0.4	-1.1	2.4
160	160	0.1 ± 0.3	-1.4	1.5	0.7 ± 0.4	-1.1	2.0	0.7 ± 0.3	-0.9	2.3	0.5 ± 0.4	-1.2	2.1

$N_D \rightarrow$		800			1600			3200			6400		
N	K	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u
10	5	13.8 ± 1.3	6.7	19.2	11.5 ± 1.0	5.2	15.8	11.5 ± 1.2	4.3	17.1	10.0 ± 1.3	2.9	14.5
10	10	12.3 ± 1.2	5.5	18.1	10.5 ± 1.2	3.9	14.3	8.4 ± 1.1	2.9	11.3	8.4 ± 1.1	2.4	11.9
20	5	7.7 ± 0.9	2.6	11.7	7.7 ± 1.0	1.8	11.8	5.9 ± 0.9	1.2	8.2	5.7 ± 1.0	0.9	7.8
20	10	8.2 ± 1.0	2.8	12.6	6.0 ± 0.9	1.6	8.5	3.4 ± 0.7	0.7	4.0	3.1 ± 0.6	0.4	3.7
20	20	8.7 ± 1.1	2.8	12.6	7.4 ± 1.0	2.0	11.2	5.8 ± 0.9	1.1	7.5	5.0 ± 0.9	0.6	6.7
40	5	4.6 ± 0.7	1.3	6.3	3.1 ± 0.6	0.7	3.9	3.7 ± 0.7	0.5	4.3	3.4 ± 0.8	0.2	3.6
40	10	4.8 ± 0.7	1.6	6.5	3.5 ± 0.6	1.0	4.4	3.1 ± 0.7	0.5	3.3	2.3 ± 0.6	0.2	2.2
40	20	6.0 ± 0.8	1.8	8.2	6.1 ± 0.9	1.6	8.8	5.4 ± 0.9	0.8	8.0	5.0 ± 1.0	0.5	7.7
40	40	7.5 ± 1.0	2.7	10.0	6.6 ± 1.0	1.7	10.0	7.3 ± 1.0	1.1	11.0	5.9 ± 1.0	0.6	9.6
80	5	2.3 ± 0.4	0.5	3.4	2.0 ± 0.5	0.3	2.7	2.2 ± 0.5	0.2	2.7	1.8 ± 0.5	0.1	1.8
80	10	2.3 ± 0.4	0.6	3.6	2.3 ± 0.4	0.5	3.2	1.9 ± 0.4	0.4	2.4	1.3 ± 0.3	0.1	1.4
80	20	2.7 ± 0.5	0.7	4.5	3.0 ± 0.5	0.8	4.4	3.9 ± 0.6	0.8	5.1	3.6 ± 0.6	0.6	5.1
80	40	2.9 ± 0.5	0.6	4.4	3.9 ± 0.6	0.9	5.6	5.4 ± 0.8	1.3	8.0	5.2 ± 0.7	1.0	7.6
80	80	3.3 ± 0.5	0.9	5.3	4.0 ± 0.6	1.0	6.3	4.9 ± 0.8	1.0	7.5	5.2 ± 0.7	0.7	7.9
160	5	1.3 ± 0.4	-0.1	2.8	0.8 ± 0.3	-0.5	1.9	0.9 ± 0.3	-0.1	1.4	0.9 ± 0.3	0.0	1.2
160	10	1.2 ± 0.3	-0.6	2.7	1.2 ± 0.3	0.0	2.6	0.8 ± 0.3	-0.2	1.7	0.9 ± 0.2	-0.1	1.3
160	20	1.2 ± 0.4	-0.7	3.0	1.2 ± 0.3	-0.4	2.6	1.7 ± 0.4	0.1	2.4	2.1 ± 0.4	0.1	3.3
160	40	1.1 ± 0.3	-0.6	2.7	1.6 ± 0.4	-0.4	2.8	2.6 ± 0.5	0.3	3.9	2.6 ± 0.5	0.3	3.9
160	80	0.7 ± 0.4	-1.2	2.3	1.6 ± 0.4	-0.4	3.2	2.2 ± 0.5	-0.2	4.0	3.0 ± 0.5	0.5	4.7
160	160	1.0 ± 0.4	-0.8	2.5	1.7 ± 0.4	0.0	3.1	2.6 ± 0.5	0.4	4.0	3.4 ± 0.5	0.6	5.5

Table 2: Exploration of NMA performance versus SNN performance as N_D, N and K are varied. The error ranges are 99% confidence intervals in the mean. Q_l and Q_u denote the lower and upper quartiles, respectively.

These experiments tested the sensitivity on the approximation parameter n and the number of nodes N. Setting $N_D = 100$, $N_{trials} = 40$, varying N over the values $\{20, 40, 80, 160\}$, varying n over the values $\{1, 5, 7, 10, 12, 14\}$, varying k over the values $\{1, 2, 3, 4\}$, and $K \leftarrow [1, \ldots, N]$. The compiled results are shown in Table 3. The ranges denote the 99% confidence interval of the mean. Table 3 shows that for a wide range of sensible values for these three parameters, we pay little

N	k	n	δ (%)
20	1	7	-0.1 ± 0.4
40	1	7	0.0 ± 0.5
80	1	7	1 ± 1
160	1	7	-2 ± 2
40	1	1	2 ± 2
40	1	5	0.3 ± 0.8
40	1	7	0.0 ± 0.5
40	1	10	0.1 ± 0.6
40	1	12	-0.1 ± 0.4
40	1	14	-0.5 ± 0.3
20	1	7	-0.1 ± 0.4
20	2	7	0.2 ± 0.9
20	3	7	1 ± 1
20	4	7	0 ± 1

Table 3: The values of δ between model averaging over $\mathcal{L}_k^\pi$ and model averaging over $\mathcal{L}_{kn}^\pi(D)$ as various parameters are varied.

classification cost by averaging over $\mathcal{L}_{kn}^\pi(D)$ versus $\mathcal{L}_k^\pi$. Especially interesting is that the difference is small even for quite small values of the approximation level n. Even for $n \gtrsim 5$ the percent remaining area captured by averaging over the full class is less than 1.1% (i.e., $0.3\% + 0.80\%$) with probability $P > 0.99$. Also, one might expect the approximation error to increase as k was increased, since for a fixed n, $\mathcal{L}_k^\pi$ includes increasingly more structures than $\mathcal{L}_{kn}^\pi(D)$ as k increases. Table 3 shows that over the range of k considered, there is not much sensitivity in the results to k. These results, although not comprehensive in scope, support that averaging over $\mathcal{L}_{kn}^\pi(D)$ does not severely degrade the ROC area relative to model averaging over $\mathcal{L}_k^\pi$. This result is important because only over $\mathcal{L}_{kn}^\pi(D)$ can we select a single tractable model to do model averaging via standard Bayesian network inference.

4.2.3 AMA VERSUS GTT, NMA AND SNN

In terms of classification accuracy, a more practical test of the benefits of AMA is to contrast its performance directly with other algorithms. In our first set of measurements to test this, we generated synthetic data and measured the performance of AMA relative to GTT, NMA and SNN. We varied N over the set $\{20, 40\}$ (much higher was too time-consuming for the complete range of k and n below), and simultaneously varied N_D from $\{100, 1000\}$, k from 2–5, n from 4–14 and $K \leftarrow \{1, \ldots, N\}$. In these experiments, N_{trials} varied from 50 to ~ 100, depending on the speed

at which the AMA model could be learned. The results for $N = 20$ and 40 are shown in Table 4 and 5, respectively. In these and all subsequent tables (except where explicitly stated), the error ranges represent 99% confidence intervals, and Q_l and Q_u denote the lower and upper quartiles, respectively.

These results illustrate several points. First is that AMA performance is relatively insensitive to the value of n. This is an encouraging result, because an approximation network with a large maximum in-degree can result in slow inference when the feature vector to be classified is not complete and N is large. Next we note the evident complementarity between the naïve classifiers (both NMA and SNN) versus GTT. The naïve classifiers perform consistently better at low N_D; whereas GTT does better at high values of N_D. This effect may be due to the difficulty of reliably extracting structural information from very small databases; in which case GTT is not able to generalize well. A final observation about these tables is that for high N_D, GTT can achieve statistically significant and large gains compared to AMA when k is sufficiently small. This effect is more important as N increases because it becomes more difficult to average over $\mathcal{L}^{\pi}_{kn}(D)$ for a fixed k as N is increased. For example, in our experiments it was too computationally costly for us to repeatedly apply AMA enough to get statistically significant measurements for ($N = 40$, $k = 5$). A similar but weaker effect is seen for the naïve classifiers. As N gets large the benefits of AMA appear to lose statistical significance at low N_D; however rarely did the naïve classifiers outperform AMA in a statistically significant sense.

Figure 4 summarizes the results of Tables 4–5 by showing the qualitative rankings of each algorithm as various parameters are varied. These rankings were derived by examining the results in Tables 4–5 and for each classifier $C_i \in \{\text{NMA, GTT, SNN }\}$, calculating $\bar{\delta}_i$ which is the average over all values of n for the particular configuration being considered in Figure 4. The following rules were applied to determine the rankings:

1. If AMA scored significantly better than C_i for a majority of runs, then AMA is ranked higher than C_i, and visa-versa.

2. If $\bar{\delta}_i < \bar{\delta}_j$ then C_i is ranked above C_j.

It is clear from this figure that the quality of AMA classifications depends strongly on both k and N_D.

While synthetic experiments are attractive because they allow us to systematically vary parameters and generate enough samples to achieve statistical significance, they do not necessary reflect performance in the real world. Also, our synthetic data generation process assumed no hidden variables, a fact which might bias our results. To this end we tested the four classifiers on 34 data sets taken from the UCI online database (Blake and Merz, 1998). These results are shown in Table 6.

Here the score δ^i_d for classifier C_i was calculated according to Procedure 3, where $M_2 = C_i$ and M_1 was taken to be the maximum scoring classifier for the data set d. For example, in the monks-2 database, AMA was the highest scoring classifier and covered 48% of the remaining area for SNN and GTT and 21% of the remaining area for NMA. The ROC area will in general depend on which state of the classification variable is considered to be the “positive" state. The scores in Table 6 are average scores for all ROC curves associated with a particular classification variable; therefore some data sets (e.g., wine) have no zero entries when two or more classifiers score highest on different curves.

Configuration			vs. NMA			vs. GTT			vs. SNN		
N_D	k	n	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u
100	2	4	3 ± 3	-4	7	9 ± 3	0	18	13 ± 3	5	19
100	2	6	6 ± 3	-2	10	9 ± 4	2	16	16 ± 4	5	26
100	2	8	5 ± 3	-2	8	8 ± 3	2	16	14 ± 3	5	20
100	2	10	7 ± 3	-1	13	10 ± 3	2	18	15 ± 4	4	24
100	2	12	5 ± 2	-2	9	11 ± 3	4	18	13 ± 3	6	20
100	2	14	7 ± 3	-1	9	11 ± 4	4	18	16 ± 3	6	24
100	3	4	4 ± 4	-4	10	8 ± 3	2	12	13 ± 4	2	21
100	3	6	6 ± 4	-2	13	12 ± 3	4	17	14 ± 4	6	23
100	3	8	9 ± 4	-2	16	13 ± 3	5	18	19 ± 4	7	27
100	3	10	8 ± 4	-2	17	11 ± 3	3	16	17 ± 4	6	24
100	3	12	7 ± 3	-1	9	10 ± 2	3	14	17 ± 3	6	28
100	3	14	6 ± 4	-3	12	12 ± 3	5	18	15 ± 4	3	22
100	4	4	5 ± 4	-4	14	8 ± 3	2	12	15 ± 4	4	22
100	4	6	5 ± 3	-2	12	9 ± 2	3	15	15 ± 3	6	24
100	4	8	5 ± 4	-2	11	10 ± 3	3	14	13 ± 4	3	23
100	4	10	7 ± 3	-2	13	12 ± 3	3	17	16 ± 3	6	23
100	4	12	4 ± 3	-3	8	9 ± 2	3	14	13 ± 3	4	21
100	4	14	8 ± 4	-1	15	12 ± 3	4	18	17 ± 3	8	27
100	5	6	7 ± 4	-3	16	10 ± 3	3	13	15 ± 4	4	23
100	5	8	5 ± 3	-3	10	10 ± 2	5	13	13 ± 3	3	21
100	5	10	4 ± 4	-2	11	9 ± 3	3	14	12 ± 4	4	20
100	5	12	6 ± 3	-2	10	12 ± 3	4	17	16 ± 3	7	24
100	5	14	6 ± 3	-2	9	10 ± 2	4	15	15 ± 3	6	19
1000	2	4	10 ± 4	-2	18	-8 ± 5	-21	10	17 ± 3	6	26
1000	2	6	11 ± 3	1	19	-12 ± 7	-26	11	19 ± 3	11	27
1000	2	8	10 ± 3	1	17	-15 ± 7	-33	7	17 ± 3	7	24
1000	2	10	12 ± 4	2	23	-8 ± 8	-17	13	20 ± 4	13	30
1000	2	12	10 ± 3	2	18	-10 ± 7	-23	10	16 ± 3	9	23
1000	2	14	10 ± 3	2	18	-7 ± 5	-21	10	18 ± 3	11	23
1000	3	4	14 ± 4	2	23	1 ± 4	-8	9	23 ± 4	11	34
1000	3	6	15 ± 4	2	26	-2 ± 5	-7	9	21 ± 3	11	31
1000	3	8	20 ± 4	6	31	2 ± 4	-6	14	27 ± 4	16	35
1000	3	10	15 ± 3	3	23	0 ± 5	-9	13	21 ± 3	13	28
1000	3	12	18 ± 4	5	29	4 ± 4	-3	14	24 ± 4	13	33
1000	3	14	18 ± 4	4	28	1 ± 4	-3	12	24 ± 3	13	33
1000	4	4	23 ± 5	5	37	6 ± 3	1	12	29 ± 4	15	42
1000	4	6	20 ± 5	2	34	8 ± 3	2	13	29 ± 4	14	41
1000	4	8	24 ± 5	5	40	8 ± 3	2	12	31 ± 4	15	45
1000	4	10	21 ± 4	8	32	6 ± 3	1	15	28 ± 3	18	37
1000	4	12	20 ± 4	8	32	7 ± 2	1	14	27 ± 3	18	37
1000	4	14	25 ± 4	7	37	8 ± 3	1	14	31 ± 4	17	45
1000	5	6	23 ± 4	6	38	9 ± 3	3	15	28 ± 4	16	39
1000	5	8	23 ± 5	6	37	10 ± 3	3	14	29 ± 4	15	42
1000	5	10	26 ± 5	9	40	9 ± 2	3	12	32 ± 4	18	45
1000	5	12	25 ± 4	12	38	10 ± 3	2	16	31 ± 4	20	42
1000	5	14	23 ± 4	8	38	9 ± 3	2	16	28 ± 4	15	42

Table 4: Exploration of AMA performance for $N = 20$ as N_D, k and n are varied. Error ranges denote the 99% confidence intervals; Q_l and Q_u denote the lower and upper quartiles, respectively.

Configuration			vs. NMA			vs. GTT			vs. SNN		
N_D	k	n	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u	δ (%)	Q_l	Q_u
100	2	4	1 ± 3	-5	5	6 ± 4	-1	9	5 ± 3	-2	9
100	2	6	3 ± 3	-2	8	5 ± 4	-2	13	6 ± 4	0	13
100	2	8	3 ± 4	-5	10	7 ± 4	-1	10	6 ± 4	-2	15
100	2	10	3 ± 5	-4	4	7 ± 4	1	10	8 ± 5	1	11
100	3	4	1 ± 5	-7	4	6 ± 3	0	6	5 ± 5	-4	8
100	3	6	1 ± 4	-5	5	5 ± 3	-2	9	5 ± 4	-3	10
100	3	8	1 ± 4	-6	4	4 ± 3	-1	9	4 ± 4	-3	9
100	3	10	3 ± 4	-4	7	8 ± 3	1	15	6 ± 4	1	9
100	4	4	-5 ± 5	-10	1	2 ± 4	-3	8	-1 ± 5	-7	5
100	4	6	0 ± 5	-7	7	4 ± 4	-1	6	3 ± 5	-3	6
100	4	8	0 ± 3	-6	5	5 ± 3	0	11	5 ± 4	-1	9
100	4	10	2 ± 5	-4	3	4 ± 2	0	7	6 ± 5	-2	10
1000	2	4	7 ± 4	0	12	-10 ± 8	-21	6	11 ± 4	3	16
1000	2	6	8 ± 4	-1	12	-8 ± 9	-22	7	14 ± 4	5	19
1000	2	8	11 ± 5	1	19	-10 ± 11	-38	13	17 ± 5	8	27
1000	2	10	5 ± 4	-2	11	-15 ± 14	-29	8	11 ± 5	3	20
1000	3	4	14 ± 5	2	26	-3 ± 6	-9	7	19 ± 5	6	29
1000	3	6	19 ± 6	6	30	0 ± 5	-6	7	23 ± 6	11	32
1000	3	8	18 ± 5	7	27	3 ± 6	-4	14	23 ± 4	12	31
1000	3	10	12 ± 5	1	18	0 ± 7	-1	11	18 ± 4	8	25
1000	4	4	19 ± 6	2	29	10 ± 4	2	17	25 ± 5	12	35
1000	4	6	20 ± 5	6	33	6 ± 4	-1	11	25 ± 5	18	35
1000	4	8	18 ± 6	1	28	9 ± 4	3	12	23 ± 5	9	29
1000	4	10	22 ± 6	1	37	8 ± 4	1	12	28 ± 5	13	42

Table 5: Exploration of AMA performance over parameter space for $N = 40$.

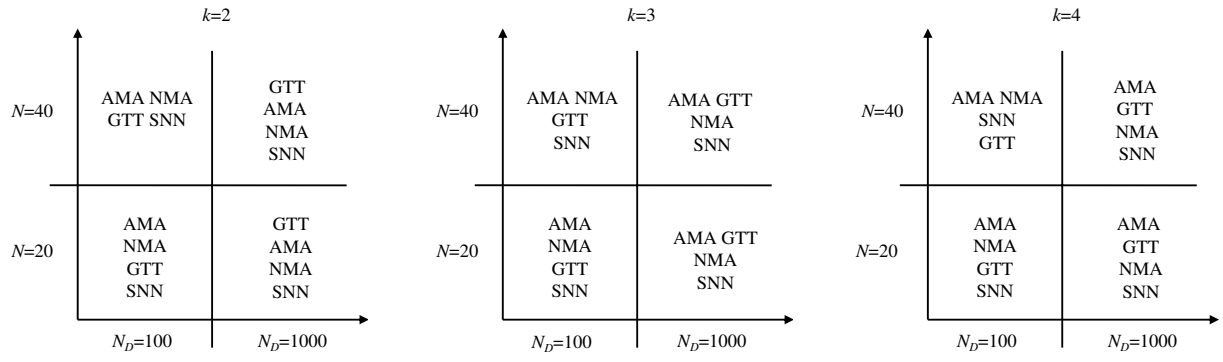


Figure 4: Qualitative comparison showing the ranking of the four algorithms as the number of nodes, the number of records and k are varied.

Data set	δ^{SNN}	δ^{GTT}	δ^{NMA}	δ^{AMA}	N	k	N_D	method
haberman	0.35	0.35	**0.00**	**0.00**	4	4	306	LOO
servo	0.56	0.66	**0.15**	**0.00**	5	5	167	LOO
lenses	0.37	0.45	**0.00**	**0.03**	6	6	24	LOO
hayes-roth	0.32	0.32	**0.00**	**0.01**	6	6	132	LOO
liver-disorders	0.14	0.07	**0.00**	**0.03**	7	7	345	LOO
monks-3	0.83	**0.24**	0.82	**0.00**	7	7	552	T&T
monks-1	0.98	**0.00**	0.98	**0.00**	7	7	554	T&T
monks-2	0.48	0.48	**0.21**	**0.00**	7	7	600	T&T
chess krkopt	0.54	**0.00**	0.54	**0.32**	7	7	28055	CV2
ecoli	0.03	**0.01**	0.02	**0.00**	8	8	336	LOO
yeast	**0.04**	0.11	**0.04**	0.07	8	8	1484	CV2
post-operative	**0.08**	0.46	**0.01**	0.09	9	9	90	LOO
prima-indian diab	**0.01**	**0.01**	**0.01**	0.02	9	9	768	CV2
abalone	0.12	0.08	**0.05**	**0.00**	9	9	4176	CV2
cpu-performance	0.13	0.31	**0.01**	**0.11**	10	10	209	CV2
glass	**0.10**	**0.04**	0.15	0.13	10	10	214	CV2
cmc	**0.01**	0.07	**0.01**	0.04	10	10	1473	CV2
sol-flare-C	0.03	0.09	**0.02**	**0.01**	11	11	322	CV2
sol-flare-M	**0.00**	0.44	**0.17**	0.20	11	11	322	CV2
sol-flare-X	**0.06**	**0.01**	0.18	0.33	11	11	322	CV2
page-blocks	0.30	**0.12**	0.23	**0.02**	11	11	5473	CV2
wine	0.14	**0.01**	0.16	**0.06**	14	7	177	CV2
heart-disease	**0.00**	**0.07**	0.11	0.19	14	6	294	CV2
housing	0.20	**0.06**	0.22	**0.00**	14	5	506	CV2
credit-screening	**0.00**	0.12	0.09	**0.02**	16	5	652	CV2
pendigits	0.58	**0.00**	0.58	**0.00**	17	6	7495	T&T
letter-recognit	0.38	**0.00**	0.38	**0.01**	17	5	20000	T&T
thyroid-disease	0.17	0.28	**0.00**	**0.11**	21	5	7200	T&T
soybean-small	**0.00**	**0.00**	**0.00**	**0.00**	22	4	47	CV4
mushroom	0.86	**0.00**	0.89	**0.03**	22	4	8124	CV2
spect	0.18	0.38	**0.16**	**0.00**	23	4	267	T&T
brst-canc-wisc	0.28	**0.00**	0.29	**0.23**	32	3	569	CV2
connect-4	**0.49**	**0.00**	**0.49**	0.52	43	2	67557	CV2
spambase	0.24	0.25	**0.00**	**0.10**	58	2	4600	CV2

Table 6: Experimental results for 34 UCI data sets. The top scoring classifier for each data set is underlined, the top two are shown in bold. AMA scored in the top one 13 of 34 times compared to 7, 12 and 12 for SNN, GTT and NMA, respectively. It scored in the top two 25 times compared to 11, 17, and 19 for NMA, GTT and SNN, respectively.

The average difference Δ^i between classifier i and AMA: $\Delta^i \equiv \frac{1}{34}\sum_d(\delta_d^{AMA} - \delta_d^i)$, was calculated to gauge the statistical significance of these experiments. The results are shown in Table 7. AMA benefits over the naïve models were significant at the 99% level, but only at the 95% level for GTT.

i	Δ^i (%)	Q_l	Q_u
SNN	19 ± 5	0	33
NMA	13 ± 5	0	23
GTT	8 ± 4	0	20

Table 7: Compiled UCI results. The error ranges denote the error of the mean.

Finally, the performance of AMA was also tested by generating training and test data with the benchmark ALARM network. In this case, $N = 36$ and $K = 4$ were fixed by the network, and a test was performed with $k = 3$, $n = 10$, and N_D systematically varied. The results in Table 8 are shown for classification on the *kinked tube (kt)* and *anaphylaxis (an)* diagnostic nodes. Here, for small number

N_D	δ^{kt} (%)	Q_l^{kt}	Q_u^{kt}	δ^{an} (%)	Q_l^{an}	Q_u^{an}
50	32 ± 13	24	55	3 ± 3	-9	17
100	23 ± 11	9	53	1 ± 3	-11	16
200	13 ± 9	-1	32	-3 ± 4	-17	16
400	12 ± 7	-1	34	-3 ± 5	-21	18
800	4 ± 7	-9	23	2 ± 5	-11	21
3200	0 ± 14	-19	15	6 ± 7	-8	19

Table 8: AMA performance v.s. GTT on synthetic data generated using the ALARM network and classifying on *kinked tube* (kt) and *anaphylaxis* (an).

of records, AMA outperformed GTT at the 99% significance level classifying on the *kinked tube* node; however, it showed no improvement when classifying on the *anaphylaxis* node. These results are notable because they demonstrate that the qualitative performance of the AMA classifier depends not just on global network features but also on features specific to the classification node. Precisely what features of the classification node are important is an open question for future research. The prior probability of *anaphylaxis* was about 4 times smaller than that of *kinked tube*; however, the local topology of the network may play a factor as well.

Obviously, using AMA was not without cost. The time to construct the models (and memory requirements) appeared to grow exponentially with k, as shown in Table 9 for $N = 40$.

5. Discussion

We have shown that, under certain assumptions, it is possible to construct a single Bayesian network model, M, whose joint distribution will be identical to exact model averaging over the class, $\mathcal{L}_k^\pi$, of models consistent with a partial ordering π and having in-degree bounded by k. Although for most partial orderings, M will be intractable to build and use for inference, we have demonstrated two ways of putting this technique to practical use: first, by constructing a single network with a particular parameterization that produces approximate model averaged predictions, and second by applying the method to the class of naïve Bayes models, leading to a simple re-parameterization

N_D	Algorithm	train time	test time
100	AMA-4	330	0.101
	AMA-3	27	0.038
	AMA-2	2.6	0.033
	GTT	0.3	0.025
	NMA	0.2	0.021
	SNN	0.02	0.017
1000	AMA-4	1000	0.113
	AMA-3	96	0.035
	AMA-2	8.9	0.033
	GTT	1.2	0.024
	NMA	0.2	0.022
	SNN	0.04	0.017

Table 9: Average training and testing times in seconds for the different algorithms with $N = 40$. "AMA-m" refers to the average over all runs with $k = m$.

that produces predictions equivalent to model averaging over all feature sets, effectively solving the feature selection problem for naïve models in a Bayesian framework.

As an example of the utility of this method, we performed some empirical studies in a classification context, and showed that on both synthetic and UCI datasets, even with relatively little effort in choosing a good value for π and with simple noninformative priors, classifications can be beneficial compared to other common BN classifiers. We have also demonstrated empirically that classifications obtained by model averaging over all naïve features sets is very likely to be beneficial over a single naïve model chosen by selecting the MAP feature set. It can be expected that these results would improve in real-world situations when expert knowledge about realistic node-orderings and structure and parameter priors can be brought to bear.

Our empirical results provide evidence that Bayesian model averaging can improve prediction over model selection, in contrast to the conclusions drawn by Domingos (2000) that Bayesian learning exacerbates the over-fitting problem. First, in our experiments with naïve classifiers, model averaging clearly produced better predictions compared to model selection in terms of the ROC area. Second, in our experiments with approximate model-averaged (AMA) classifiers, we observed the trend that AMA classifications performed successively better as more and more structures were included in the model averaging (i.e., as k was increased). This conflicts with the assertion that model generalization suffers when more models are considered in the averaging process.

In general, the benefits of AMA were not without cost. Construction times for AMA models were higher than other model types, and were observed empirically to grow exponentially as the maximum in-degree k increased. Furthermore, when the approximation parameter n is large, inference with incomplete feature vectors can become prohibitive. The latter observation is mitigated by the fact that the AMA classifier is generally insensitive to the value of n, allowing n to be minimized without sacrificing classification accuracy. However, in cases where the number of nodes, N, is very large, the cost of building the AMA classifier might outweigh the benefits: in this case, if the number of records, N_D, is small then a naïve model-averaged (NMA) classifier would probably be

the most attractive since it performed comparably to AMA with orders of magnitude faster training time.

Once the model-averaging model is built, however, the technique has an advantage because of its simplicity of implementation. Existing systems that use Bayesian network classifiers can trivially be adapted to use model averaging by replacing their existing model with a single summary model. This is especially relevant in cases where a naïve classifier is currently being employed, as building a NMA classifier retains the same linear time and space complexity required for building a naïve model.

As already stated, when n is large enough the approximation network can be extremely dense, thus making inference difficult when the feature-vector is incomplete. One way to get around this issue, when the incompleteness of the feature-vector is regular, is to learn separate model-averaging models on subsets of data in which the same set of features is missing. Thus a one-time investment of building several feature-vector-specific models would allow us to do $O(N)$ inference even for large n-values.

Future work includes finding a better method for optimizing the ordering π, possibly by doing a search over orderings as in (Friedman and Koller, 2003), and perhaps using cached sufficient statistics with advanced data structures such as ADTrees (Moore and Lee, 1998) to increase the practical limits of k. There are a wealth of other classifiers that it would be interesting to compare with our approach: both non-probabilistic based models such as C4.5, neural networks, support-vector machines, etc., and other model-averaging techniques such as those presented in Madigan and Raftery (1994) and Cerquides (2003)

It should also be possible to relax the assumption of complete training data by using the EM algorithm or MCMC sampling to estimate parameters from data. Finally, the identification of other classes of BN models that easily fit within the $\mathcal{L}_k^\pi$ class could lead to other especially efficient solutions such as that obtained with the naïve Bayes model.

Acknowledgments

We would like to acknowledge the reviewers for their very thorough comments of this paper. This research was supported in part by grant RO1-GM61992-01A1 from the National Institutes of Health and by grant IIS-0325581 from the National Science Foundation.

References

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

W. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–91)*, pages 52–60, San Mateo, California, 1991. Morgan Kaufmann Publishers.

Jesús Cerquides. *Improving Bayesian network classifiers*. PhD thesis, Technical University of Catalonia, 2003.

Jesús Cerquides and López de Màntaras. Tractable Bayesian learning of tree augmented naive Bayes classifiers. *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2000)*, pages 75–82, 2003.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

Denver Dash and Gregory F. Cooper. Exact model averaging with naive Bayesian classifiers. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)*, pages 91–98, January 2002.

Denver Dash and Gregory F. Cooper. Exact model averaging with discrete Bayesian classifiers. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, Florida, 2003.

Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 223–230, 2000.

Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

Marek J. Druzdzel. SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: A development environment for graphical decision-theoretic models. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI–99)*, pages 902–903, Orlando, FL, July 18–22 1999.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, NY, 1973.

J. P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press, New York, New York, 1975.

Jerome Friedman. On bias, variance, 0/1–loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2–3):131–163, 1997.

Nir Friedman and Daphne Koller. Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1–2):95–125, 2003.

David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97 (1-2):273–324, 1997. URL `citeseer.nj.nec.com/kohavi96wrappers.html`.

Pat Langley and Stephanie Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 399–406, San Francisco, CA, 1994. Morgan Kaufmann Publishers.

Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B (Methodological)*, 50(2):157–224, 1988.

David Madigan and Adrian E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89: 1535–1546, 1994.

David Madigan and J. York. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232, 1995.

Marina Meila and Tommi S. Jaakkola. Tractable Bayesian learning of tree belief networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)*, pages 380–388, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

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 Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 841–848, Cambridge, MA, 2002. MIT Press.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer Verlag, New York, 1993.

T. S. Verma and Judea Pearl. Equivalence and synthesis of causal models. In P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pages 255 –269. Elsevier Science Publishing Company, Inc., New York, N. Y., 1991.

C. T. Volinsky. *Bayesian Model Averaging for Censored Survival Models*. PhD dissertation, University of Washington, 1997.

Journal of Machine Learning Research 5 (2004) 1205–1224 Submitted 1/04; Revised 5/04; Published 10/04

Efficient Feature Selection via Analysis of Relevance and Redundancy

Lei Yu LEIYU@ASU.EDU
Huan Liu HLIU@ASU.EDU
Department of Computer Science & Engineering
Arizona State University
Tempe, AZ 85287-8809, USA

Editor: Isabelle Guyon

Abstract

Feature selection is applied to reduce the number of features in many applications where data has hundreds or thousands of features. Existing feature selection methods mainly focus on finding relevant features. In this paper, we show that feature relevance alone is insufficient for efficient feature selection of high-dimensional data. We define feature redundancy and propose to perform explicit redundancy analysis in feature selection. A new framework is introduced that decouples relevance analysis and redundancy analysis. We develop a correlation-based method for relevance and redundancy analysis, and conduct an empirical study of its efficiency and effectiveness comparing with representative methods.

Keywords: supervised learning, feature selection, relevance, redundancy, high dimensionality

1. Introduction

In classic supervised learning, one is given a training set of labeled fixed-length feature vectors (instances). An instance is typically described as an assignment of values $f = (f_1, ..., f_N)$ to a set of features $F = (F_1, ..., F_N)$ and one of l possible classes $c_1, ..., c_l$ to the class label C. The task is to induce a hypothesis (classifier) that accurately predicts the labels of novel instances. The learning of the classifier is inherently determined by the feature-values. In theory, more features should provide more discriminating power, but in practice, with a limited amount of training data, excessive features will not only significantly slow down the learning process, but also cause the classifier to over-fit the training data as irrelevant or redundant features may confuse the learning algorithm.

Feature selection has been an active and fruitful field of research and development for decades in statistical pattern recognition (Mitra et al., 2002), machine learning (Liu et al., 2002b; Robnik-Sikonja and Kononenko, 2003), data mining (Kim et al., 2000; Dash et al., 2002) and statistics (Hastie et al., 2001; Miller, 2002). It has proven in both theory and practice effective in enhancing learning efficiency, increasing predictive accuracy, and reducing complexity of learned results (Almuallim and Dietterich, 1994; Koller and Sahami, 1996; Blum and Langley, 1997). Let G be some subset of F and f_G be the value vector of G. In general, the goal of feature selection can be formalized as selecting a minimum subset G such that $\mathbf{P}(C \mid G = f_G)$ is equal or as close as possible to $\mathbf{P}(C \mid F = f)$, where $\mathbf{P}(C \mid G = f_G)$ is the probability distribution of different classes given the feature values in G and $\mathbf{P}(C \mid F = f)$ is the original distribution given the feature values in F (Koller and Sahami, 1996). We call such a minimum subset an *optimal* subset, illustrated by the example below.

Example 1 (Optimal subset) *Let features $F_1, ..., F_5$ be Boolean. The target concept is $C = g(F_1, F_2)$ where g is a Boolean function. With $F_2 = \overline{F_3}$ and $F_4 = \overline{F_5}$, there are only eight possible instances. In order to determine the target concept, F_1 is indispensable; one of F_2 and F_3 can be disposed of (note that C can also be determined by $g(F_1, \overline{F_3})$), but we must have one of them; both F_4 and F_5 can be discarded. Either $\{F_1, F_2\}$ or $\{F_1, F_3\}$ is an optimal subset. The goal of feature selection is to find either of them.*

In the presence of hundreds or thousands of features, researchers notice (Yang and Pederson, 1997; Xing et al., 2001) that it is common that a large number of features are not informative because they are either irrelevant or redundant with respect to the class concept. In other words, learning can be achieved more efficiently and effectively with just relevant and non-redundant features. However, the number of possible feature subsets grows exponentially with the increase of dimensionality. Finding an optimal subset is usually intractable (Kohavi and John, 1997) and many problems related to feature selection have been shown to be NP-hard (Blum and Rivest, 1992).

Researchers have studied various aspects of feature selection. One of the key aspects is to measure the *goodness* of a feature subset in determining an optimal one (Liu and Motoda, 1998). Different feature selection methods can be broadly categorized into the *wrapper* model (Kohavi and John, 1997; Kim et al., 2000) and the *filter* model (Liu and Setiono, 1996; Liu et al., 2002b; Hall, 2000; Yu and Liu, 2003). The wrapper model uses the predictive accuracy of a predetermined learning algorithm to determine the goodness of the selected subsets. These methods are computationally expensive for data with a large number of features (Kohavi and John, 1997). The filter model separates feature selection from classifier learning and selects feature subsets that are independent of any learning algorithm. It relies on various measures of the general characteristics of the training data such as distance, information, dependency, and consistency (Liu and Motoda, 1998). *Search* is another key problem in feature selection. To balance the tradeoff of result optimality and computational efficiency, different search strategies such as complete, heuristic, and random search have been studied to generate candidate feature subsets for evaluation (Blum and Langley, 1997; Dash and Liu, 2003). According to the availability of class labels, there are feature selection methods for *supervised learning* (Dash and Liu, 1997; Yu and Liu, 2003) as well as for *unsupervised learning* (Kim et al., 2000; Dash et al., 2002). Feature selection has found success in many applications like text categorization (Yang and Pederson, 1997; Forman, 2003), image retrieval (Swets and Weng, 1995; Dy et al., 2003), genomic microarray analysis (Xing et al., 2001; Yu and Liu, 2004), customer relationship management (Ng and Liu, 2000), and intrusion detection (Lee et al., 2000).

Despite the impressive achievements in the current field of feature selection, we observe great challenges arising from domains such as genomic microarray analysis and text categorization where data may contain tens of thousands of features (Yu and Liu, 2004; Forman, 2003). First of all, the nature of high dimensionality of data can cause the so-called problem of "curse of dimensionality" (Hastie et al., 2001). Secondly, high-dimensional data often contains many redundant features. Both theoretical analysis and empirical evidence show that along with irrelevant features, redundant features also affect the speed and accuracy of learning algorithms and thus should be eliminated as well (Koller and Sahami, 1996; Kohavi and John, 1997; Hall, 2000). Existing feature selection methods mainly exploit two approaches: individual (feature) evaluation and subset evaluation (Blum and Langley, 1997; Guyon and Elisseeff, 2003). Methods of individual evaluation rank features according to their importance in differentiating instances of different classes and can only remove irrelevant features as redundant features likely have similar rankings. Methods of subset evaluation search for a minimum subset of features that satisfies some goodness measure and can remove

irrelevant features as well as redundant ones. However, among existing heuristic search strategies for subset evaluation, even greedy sequential search which reduces the search space from $O(2^N)$ to $O(N^2)$ can become very inefficient for high-dimensional data. The limitations of existing research clearly suggest that we should pursue a different framework of feature selection that allows efficient analysis of both feature relevance and redundancy for high-dimensional data.

The remainder of this paper is organized as follows. In Section 2, we review notions of feature relevance, identify the need for redundancy analysis, and provide a formal definition of feature redundancy. In Section 3, we analyze in detail the limitations of current approaches and propose a new framework of efficient feature selection. In Section 4, we describe correlation measures, and present a correlation-based method for efficient relevance and redundancy analysis under the new framework. Section 5 contains an empirical study of our method in terms of efficiency and effectiveness comparing with representative methods. Section 6 concludes this work and points out some future directions.

2. Feature Relevance and Feature Redundancy

Traditionally, feature selection research has focused on searching for relevant features. Although some recent work has pointed out the existence and effect of feature redundancy (Koller and Sahami, 1996; Kohavi and John, 1997; Hall, 2000), there is little work on explicit treatment of feature redundancy. In the following, we first present a classic notion of feature relevance and illustrate why it alone cannot handle feature redundancy, and then provide our formal definition of feature redundancy which paves the way for efficient elimination of redundant features.

2.1 Feature Relevance

Based on a review of previous definitions of feature relevance, John, Kohavi, and Pfleger classified features into three disjoint categories, namely, strongly relevant, weakly relevant, and irrelevant features (John et al., 1994). Let F be a full set of features, F_i a feature, and $S_i = F - \{F_i\}$. These categories of relevance can be formalized as follows.

Definition 1 (Strong relevance) *A feature F_i is strongly relevant iff*

$$\mathbf{P}(C \mid F_i,\ S_i) \neq \mathbf{P}(C \mid S_i)\ .$$

Definition 2 (Weak relevance) *A feature F_i is weakly relevant iff*

$$\mathbf{P}(C \mid F_i,\ S_i) = \mathbf{P}(C \mid S_i),\ \texttt{and}$$

$$\exists\ S'_i \subset S_i,\ \textit{such that}\ \mathbf{P}(C \mid F_i,\ S'_i) \neq \mathbf{P}(C \mid S'_i)\ .$$

Corollary 1 (Irrelevance) *A feature F_i is irrelevant iff*

$$\forall\ S'_i \subseteq S_i,\ \mathbf{P}(C \mid F_i,\ S'_i) = \mathbf{P}(C \mid S'_i)\ .$$

Strong relevance of a feature indicates that the feature is always necessary for an optimal subset; it cannot be removed without affecting the original conditional class distribution. Weak relevance suggests that the feature is not always necessary but may become necessary for an optimal subset at certain conditions. Irrelevance (following Definitions 1 and 2) indicates that the feature is not necessary at all. According to these definitions, it is clear that in previous Example 1, feature F_1 is strongly relevant, F_2, F_3 weakly relevant, and F_4, F_5 irrelevant. An optimal subset should include all strongly relevant features, none of irrelevant features, and a subset of weakly relevant features. However, it is not given in the definitions which of weakly relevant features should be selected and which of them removed. Therefore, it is necessary to define feature redundancy among relevant features.

2.2 Defining Feature Redundancy

Notions of feature redundancy are normally in terms of feature correlation. It is widely accepted that two features are redundant to each other if their values are completely correlated (for example, features F_2 and F_3 in Example 1). In reality, it may not be so straightforward to determine feature redundancy when a feature is correlated (perhaps partially) with a set of features. We now formally define feature redundancy in order to devise an approach to explicitly identify and eliminate redundant features. Before we proceed, we first introduce the definition of a feature's Markov blanket given by Koller and Sahami (1996).

Definition 3 (Markov blanket) *Given a feature F_i, let $M_i \subset F$ $(F_i \notin M_i)$, M_i is said to be a Markov blanket for F_i iff*

$$\mathbf{P}(F - M_i - \{F_i\},\, C \mid F_i,\, M_i) = \mathbf{P}(F - M_i - \{F_i\},\, C \mid M_i)\ .$$

The Markov blanket condition requires that M_i subsume not only the information that F_i has about C, but also about all of the other features. It is pointed out in Koller and Sahami (1996) that an optimal subset can be obtained by a backward elimination procedure, known as *Markov blanket filtering*: let G be the current set of features ($G = F$ in the beginning), at any phase, if there exists a Markov blanket for F_i within the current G, F_i is removed from G. It is proved that this process guarantees a feature removed in an earlier phase will still find a Markov blanket in any later phase, that is, removing a feature in a later phase will not render the previously removed features necessary to be included in the optimal subset. According to previous definitions of feature relevance, we can also prove that strongly relevant features cannot find any Markov blanket. Since irrelevant features should be removed anyway, we exclude them from our definition of redundant features.

Definition 4 (Redundant feature) *Let G be the current set of features, a feature is redundant and hence should be removed from G iff it is weakly relevant and has a Markov blanket M_i within G .*

From the property of Markov blanket, it is easy to see that a redundant feature removed earlier remains redundant when other features are removed. Figure 1 depicts the relationships between definitions of feature relevance and redundancy introduced so far. It shows that an entire feature set can be conceptually divided into four basic disjoint parts: irrelevant features (I), redundant features (II, part of weakly relevant features), weakly relevant but non-redundant features (III), and strongly relevant features (IV). An optimal subset essentially contains all the features in parts III and IV. It is worthy to point out that although parts II and III are disjoint, different partitions of them can result from the process of Markov blanket filtering. In previous Example 1, either of F_2 or F_3, but not both, should be removed as a redundant feature.

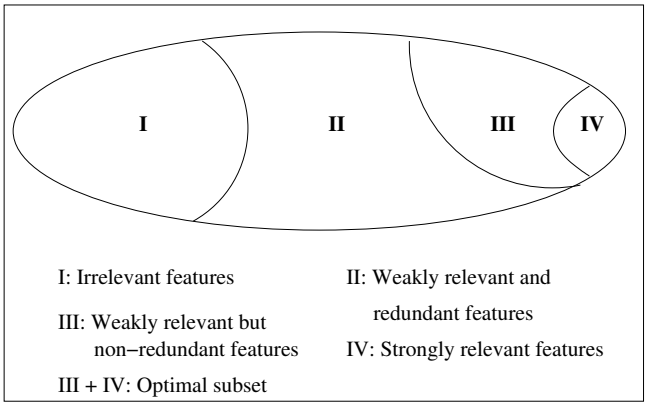


Figure 1: A view of feature relevance and redundancy.

3. Efficient Feature Selection via Relevance and Redundancy Analysis

We now review two major approaches in dealing with feature relevance and redundancy, analyze their limitations for high-dimensional data, and then propose a new framework of efficient feature selection based on relevance and redundancy analysis.

3.1 Existing Approaches in Dealing with Relevance and Redundancy

As mentioned earlier, there exist two major approaches in feature selection: *individual evaluation* and *subset evaluation*. Individual evaluation, also known as feature weighting/ranking (Blum and Langley, 1997; Guyon and Elisseeff, 2003), assesses individual features and assigns them weights according to their degrees of relevance. A subset of features is often selected from the top of a ranking list, which approximates the set of relevant features (II, III, and IV in Figure 1). With its linear time complexity in terms of dimensionality N, this approach is efficient for high-dimensional data. However, it is incapable of removing redundant features because redundant features likely have similar rankings. As long as features are deemed relevant to the class, they will all be selected even though many of them are highly correlated to each other. For high-dimensional data which may contain a large number of redundant features, this approach may produce results far from optimal.

Many feature selection methods take the subset evaluation approach which handles feature redundancy with feature relevance. The diagram in Figure 2 exhibits a traditional framework of feature selection via subset evaluation (Liu and Motoda, 1998). Subset generation produces candidate feature subsets based on a certain search strategy. Each candidate subset is evaluated by a certain evaluation measure and compared with the previous best one with respect to this measure. If a new subset turns out to be better, it replaces the previous best subset. The process of subset generation and evaluation is repeated until a given stopping criterion is satisfied. Distinguished from individual evaluation, evaluation measures used by this approach are defined against feature subsets, taking into account the existence and effect of redundant features. A feature subset selected by this approach approximates the optimal subset (parts III and IV in Figure 1). Many methods have proven effective to some extent in removing both irrelevant features and redundant features (John et al., 1994; Koller and Sahami, 1996; Bell and Wang, 2000; Hall, 2000). However, methods in this framework can suffer from an inevitable problem caused by searching through feature sub-

sets required in the subset generation step. Although there exist various heuristic search strategies such as greedy sequential search, best-first search, and genetic algorithm (Liu and Motoda, 1998), most of them still incur time complexity $O(N^2)$, which prevents them from scaling well to data sets containing tens of thousands of features.

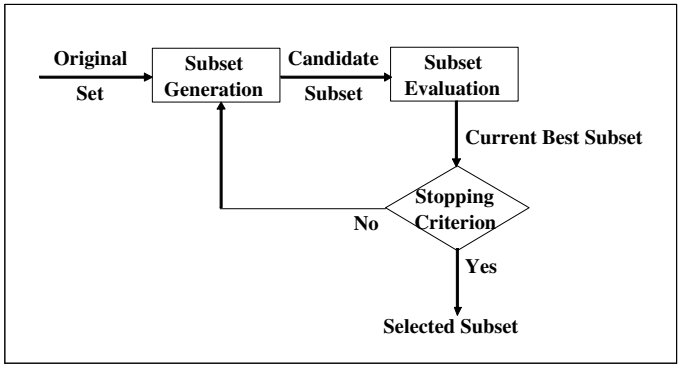


Figure 2: A traditional framework of feature selection.

3.2 A New Framework of Efficient Feature Selection

From previous discussions, it is clear that in order to eliminate redundant features, the state-of-the-art feature selection methods have to rely on the approach of subset evaluation which *implicitly* handles feature redundancy with feature relevance. These methods can produce better results than methods without handling feature redundancy, but the high computational cost of the subset search makes them inefficient for high-dimensional data. Therefore, in our solution, we propose a new framework of feature selection which avoids implicitly handling feature redundancy and turns to efficient elimination of redundant features via *explicitly handling feature redundancy*.

Relevance definitions divide features into strongly relevant, weakly relevant, and irrelevant ones; redundancy definition further divides weakly relevant features into redundant and non-redundant ones. Our goal is to efficiently find the optimal subset (parts III and IV in Figure 1). We can achieve this goal through a new framework of feature selection (shown in Figure 3) composed of two steps: first, relevance analysis determines the subset of relevant features by removing irrelevant ones, and second, redundancy analysis determines and eliminates redundant features from relevant ones and thus produces the final subset. Its advantage over the traditional framework of subset evaluation lies in that by decoupling relevance and redundancy analysis, it circumvents subset search and allows a both efficient and effective way in finding a subset that approximates an optimal subset.

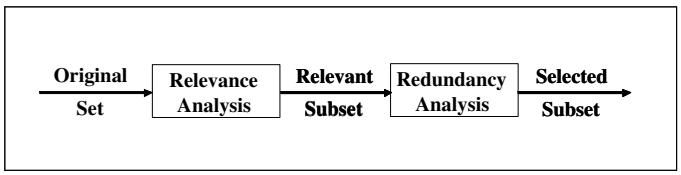


Figure 3: A new framework of feature selection.

It is sensible to use efficient heuristic methods to approximate the computation of relevant features and redundant features under our new framework for two reasons. On one hand, searching for an optimal subset based on the definitions of feature relevance and redundancy is combinatorial in nature. It is obvious that exhaustive or complete search is prohibitive with a large number of features. On the other hand, an optimal subset is defined based on the full population where the true data distribution is known. It is generally assumed (Mitchell, 1997; Miller, 2002) that a training data set is only a small portion of the full population, especially in a high-dimensional space. Therefore, it is not proper to search for an optimal subset from the training data as over-searching the training data can cause over-fitting (Jensen and Cohen, 2000). We next present our approximation method.

4. A Correlation Based Method

Correlation is widely used in machine learning and statistics for relevance analysis. In this section, we first introduce our choice of correlation measure in Section 4.1, then describe our correlation-based method for both relevance and redundancy analysis in Section 4.2, and present and analyze the algorithm in Section 4.3.

4.1 Correlation Measures

There exist broadly two types of measures for the correlation between two random variables: linear and non-linear. Of linear correlation, the most well known measure is *linear correlation coefficient*. For a pair of variables (X, Y), the linear correlation coefficient ρ is given by

$$\rho = \frac{\sum_i (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_i (x_i - \overline{x_i})^2}\sqrt{\sum_i (y_i - \overline{y_i})^2}} ,$$

where $\overline{x_i}$ is the mean of X, and $\overline{y_i}$ is the mean of Y. The value of ρ lies between -1 and 1, inclusive. If X and Y are completely correlated, ρ takes the value of 1 or -1; if X and Y are independent, ρ is zero. It is a symmetrical measure for two variables. Other measures in this category are basically variations of the above formula, such as *least square regression error* and *maximal information compression index* (Mitra et al., 2002). However, it is not safe to always assume linear correlation between features in the real world. Linear correlation measures may not be able to capture correlations that are not linear in nature. It can also be observed that linear correlation coefficient is not suitable for nominal data.

Among non-linear correlation measures, many measures are based on the information-theoretical concept of *entropy*, a measure of the uncertainty of a random variable. The entropy of a variable X is defined as

$$H(X) = -\sum_i P(x_i)\log_2(P(x_i)) ,$$

and the entropy of X after observing values of another variable Y is defined as

$$H(X|Y) = -\sum_j P(y_j)\sum_i P(x_i \mid y_j)\log_2(P(x_i \mid y_j)) ,$$

where $P(x_i)$ is the prior probabilities for all values of X, and $P(x_i \mid y_i)$ is the posterior probabilities of X given the values of Y. The amount by which the entropy of X decreases reflects additional

information about X provided by Y and is called *information gain* (Quinlan, 1993), given by

$$IG(X \mid Y) = H(X) - H(X \mid Y) .$$

According to this measure, feature Y is regarded more correlated to feature X than to feature Z, if $IG(X \mid Y) > IG(Z \mid Y)$. It is easy to prove that information gain is a symmetrical measure. Symmetry is a desired property for a measure of correlations between features. It ensures that the order of two features ((X, Y) or (Y, X)) will not affect the value of the measure. Since information gain tends to favor features with more values, it should be normalized with their corresponding entropy. Therefore, we choose *symmetrical uncertainty* (Press et al., 1988), defined as

$$SU(X, Y) = 2\left[\frac{IG(X \mid Y)}{H(X)+H(Y)}\right] .$$

It compensates for information gain's bias toward features with more values and restricts its values to the range $[0,1]$. A value of 1 indicates that knowing the values of either feature completely predicts the values of the other; a value of 0 indicates that X and Y are independent. In addition, it still treats a pair of features symmetrically. Entropy-based measures handle nominal or discrete features, and therefore continuous features need to be properly discretized (Liu et al., 2002a) in order to use entropy-based measures.

4.2 Efficient Relevance and Redundancy Analysis

Using symmetrical uncertainty (*SU*) as the correlation measure, we are ready to develop an approximation method for both relevance and redundancy analysis under our new framework introduced in Section 3.2. We first differentiate two types of correlation between features (including the class).

Definition 5 (C-correlation) *The correlation between any feature F_i and the class C is called C-correlation, denoted by $SU_{i,c}$.*

Definition 6 (F-correlation) *The correlation between any pair of features F_i and F_j $(i \neq j)$ is called F-correlation, denoted by $SU_{i,j}$.*

Aiming to achieve high efficiency, we calculate C-correlation for each feature, and heuristically decide a feature F_i to be relevant if it is highly correlated with the class C, i.e., if $SU_{i,c} > \gamma$, where γ is a relevance threshold which can be determined by users. The selected relevant features are then subject to redundancy analysis. Similarly, we can evaluate the correlation between individual features for redundancy analysis without considering the correlation between various feature subsets. However, there are two difficulties in determining feature redundancy via pair-wise F-correlation calculation: (1) when two features are not completely correlated with each other, it may be hard to determine feature redundancy and which one to be removed; and (2) it may still require F-correlation calculation for a total of $\frac{N(N-1)}{2}$ pairs, which is inefficient for high-dimensional data. Below, we try to efficiently determine feature redundancy by substantially reducing the number of feature pairs evaluated for F-correlation.

In Section 2, we apply Markov blankets to *exactly* determine feature redundancy. When it comes to *approximately* determine feature redundancy, the key is to find approximate Markov blankets for the selected relevant features. We assume that a feature with a larger C-correlation value contains

by itself more information about the class than a feature with a smaller C-correlation value. We determine the existence of an approximate Markov blanket between a pair of correlated features F_i and F_j based on their F-correlation level $SU_{i,j}$ as follows. When $SU_{j,c} \geq SU_{i,c}$, we choose to evaluate whether feature F_j can form an approximate Markov blanket for feature F_i (instead of F_i for F_j) in order to maintain more information about the class. In addition, we heuristically use $SU_{i,c}$ as a threshold to determine whether the F-correlation $SU_{i,j}$ is strong or not. An approximate Markov blanket can be defined as follows.

Definition 7 (Approximate Markov blanket) *For two relevant features F_i and F_j $(i \neq j)$, F_j forms an approximate Markov blanket for F_i iff $SU_{j,c} \geq SU_{i,c}$ and $SU_{i,j} \geq SU_{i,c}$.*

Recall that Markov blanket filtering, a backward elimination procedure based on a feature's Markov blanket in the current set, guarantees that a redundant feature removed in an earlier phase will still find a Markov blanket in any later phase when another redundant feature is removed. It is easy to verify that this is not the case for backward elimination based on a feature's approximate Markov blanket in the current set. For instance, if F_j is the only feature that forms an approximate Markov blanket for F_i, and F_k forms an approximate Markov blanket for F_j, after removing F_i based on F_j, further removing F_j based on F_k will result in no approximate Markov blanket for F_i in the current set. However, we can avoid this situation by removing a feature only when it can find an approximate Markov blanket formed by a predominant feature, defined as follows.

Definition 8 (Predominant feature) *A relevant feature is predominant iff it does not have any approximate Markov blanket in the current set.*

Predominant features will not be removed at any stage. If a feature F_i is removed based on a predominant feature F_j in an earlier phase, it is guaranteed that it will still find an approximate Markov blanket (the same F_j) in any later phase when another feature is removed. To summarize, our method for redundancy analysis consists of (1) selecting a predominant feature, (2) removing all features for which it forms an approximate Markov blanket, and (3) iterating steps (1) and (2) until no more predominate features can be selected. An optimal subset can therefore be approximated by a set of predominant features.

4.3 Algorithm and Analysis

The approximation method for relevance and redundancy analysis presented before can be realized by an algorithm, named FCBF (Fast Correlation-Based Filter). It involves two connected steps: (1) selecting a subset of relevant features, and (2) selecting predominant features from relevant ones. As shown in Figure 4, for a data set S with N features and class C, the algorithm finds a set of predominant features S_{best}. In the first step (lines 2-7), it calculates the SU value for each feature, selects relevant features into S'_{list} based on a predefined threshold δ, and orders them in a descending order according to their SU values. In the second step (lines 8-18), it further processes the ordered list S'_{list} to select predominant features. A feature F_j that has already been determined to be a predominant feature can always be used to filter out other features for which F_j forms an approximate Markov blanket. Since the feature with the highest C-correlation does not have any approximate Markov blanket, it must be one of the predominant features. So the iteration starts from the first element in S'_{list} (line 8) and continues as follows. For all the remaining features (from the one right next to F_j to the last one in S'_{list}), if F_j happens to form an approximate Markov blanket

```
input:  S(F_1, F_2, ..., F_N, C)   // a training data set
        δ                          // a predefined threshold
output: S_best                     // a selected subset

1   begin
2     for i = 1 to N do begin
3       calculate SU_{i,c} for F_i;
4       if (SU_{i,c} > δ)
5         append F_i to S'_list;
6     end;
7     order S'_list in descending SU_{i,c} value;
8     F_j = getFirstElement(S'_list);
9     do begin
10      F_i = getNextElement(S'_list, F_j);
11      if (F_i <> NULL)
12        do begin
13          if (SU_{i,j} ≥ SU_{i,c})
14            remove F_i from S'_list;
15          F_i = getNextElement(S'_list, F_i);
16        end until (F_i == NULL);
17      F_j = getNextElement(S'_list, F_j);
18    end until (F_j == NULL);
19  S_best = S'_list;
20  end;
```

Figure 4: FCBF Algorithm.

for F_i (line 13), F_i will be removed from S'_{list}. After one round of filtering features based on F_j, the algorithm will take the remaining feature right next to F_j as the new reference (line 17) to repeat the filtering process. The algorithm stops until no more predominant features can be selected. Figure 5 illustrates how predominant features are selected with the rest features removed as redundant ones. In Figure 5, six features are selected as relevant ones and ranked according to their C-correlation values, with F_1 being the most relevant one. In the first round, F_1 is selected as a predominant feature, and F_2 and F_4 are removed based on F_1. In the second round, F_3 is selected, and F_6 is removed based on F_3. In the last round, F_5 is selected.

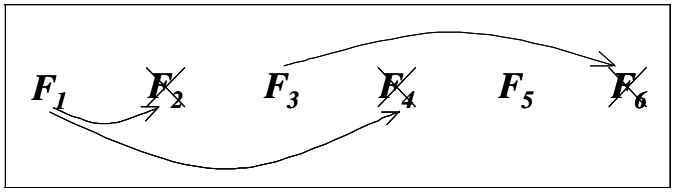


Figure 5: Selection of predominant features

We now analyze the time complexity of FCBF before an empirical study of its efficiency. As we can see from Figure 4, major computation of the algorithm involves SU values for C- and F-correlations, which has linear complexity in term of the number of instances in a data set. In terms of dimensionality N, to determine relevant features, the algorithm has linear complexity $O(N)$; to

determine predominant features from relevant ones (assuming all features are selected as relevant ones), it has a best-case complexity $O(N)$ when only one feature is selected and all of the rest of the features are removed, and a worse-case complexity $O(N^2)$ when all features are selected. These time complexity results are comparable to subset evaluation based on greedy sequential search in which features are, one at a time, added to the current subset (i.e., sequential forward selection) or removed from the current subset (i.e., sequential backward elimination). However, in general cases when k $(1 < k < N)$ features are selected, the number of evaluations performed by FCBF will typically be much less (and certainly never more) than the number of evaluations performed by greedy sequential search because features removed in each round are not considered in the next round and FCBF typically removes a large number of features (instead of only one by greedy sequential search) in each round. This makes FCBF substantially faster than algorithms of subset evaluation based on greedy sequential search, as will be demonstrated by the running time comparisons reported in Section 5. The more features removed in an earlier round, the faster FCBF is. Moreover, selecting a subset of relevant features in the first step can further improves its efficiency.

In summary, our method approximates relevance and redundancy analysis by selecting all predominant features and removing the rest features. It uses both C- and F-correlations to determine feature redundancy and combines sequential forward selection with elimination so that it not only circumvents full pair-wise F-correlation analysis but also achieves higher efficiency than pure sequential forward selection or backward elimination. However, our method is suboptimal due to the way C- and F-correlations are used for relevance and redundancy analysis and the approximates that it uses. It is fairly straightforward to improve the optimality of the results by considering different combinations of features in evaluating feature relevance and redundancy, which in turn increases time complexity. Another way to improve result optimality is to find better heuristics in determining a feature's approximate Markov blanket.

5. Empirical Study

In this section, we empirically evaluate the efficiency and effectiveness of our method by comparing FCBF with representative feature selection algorithms. We describe experimental setup in Section 5.1, discuss results on synthetic data and benchmark data in Sections 5.2 and 5.3 respectively, and summarize the findings in Section 5.4.

5.1 Experimental Setup

The efficiency of a feature selection algorithm can be directly measured by its running time over various data sets. As to effectiveness, a simple and direct evaluation criterion is how similar the selected subset and the optimal subset are, but it can only be measured over synthetic data for which we know beforehand which features are irrelevant or redundant. For real-world data, we often do not have such prior knowledge about the optimal subset, so we use the predictive accuracy on the selected subset of features as an indirect measure.

In terms of the above criteria, we limit our comparisons to the filter model as FCBF is a filter algorithm designed for high-dimensional data. We choose representative algorithms from both approaches (i.e., individual evaluation and subset evaluation). One algorithm, from individual evaluation, is ReliefF (Robnik-Sikonja and Kononenko, 2003) which searches for nearest neighbors of instances of different classes and weights features according to how well they differentiate instances of different classes. Another algorithm, from subset evaluation, is a variation of CFS (Hall, 2000),

denoted by CFS-SF (Sequential Forward). CFS exploits best-first search based on some correlation measure which evaluates the goodness of a subset by considering the individual predictive ability of each feature and the degree of correlation between them. Sequential forward selection is used in CFS-SF as initial experiments show CFS-SF runs much faster to produce similar results than CFS. A third one, also from subset evaluation, is a variation of FOCUS (Almuallim and Dietterich, 1994), denoted by FOCUS-SF. FOCUS exhaustively examines all subsets of features, selecting the minimal subset that separates classes as consistently as the full set can. It is prohibitively costly even for data sets with moderate dimensionality. FOCUS-SF replaces exhaustive search in FOCUS with sequential forward selection. In our experiments, we heuristically set the relevance threshold γ to be the SU value of the $\lfloor N/\log N \rfloor$th ranked feature for each data set. To test how the selection of threshold affects the performance of FCBF, we also include in our comparisons the results of FCBF with γ set to the default value 0. We use $\text{FCBF}_{(\log)}$ to represent a version of FCBF with the former setting, and $\text{FCBF}_{(0)}$ with the latter setting in the rest of the paper.

In addition to feature selection algorithms, we use two learning algorithms, NBC (Witten and Frank, 2000) and C4.5 (Quinlan, 1993), to evaluate the predictive accuracy on the selected subset of features. All these selected algorithms are from Weka's collection (Witten and Frank, 2000). FCBF is also implemented in Weka environment.

5.2 Results and Discussions on Synthetic Data

We use three synthetic data sets to illustrate the strengthes and limitations of FCBF and compare it with ReliefF, CFS-SF, and FOCUS-SF. The first data set is the widely used Corral data (John et al., 1994) which contains six Boolean features ($A0$, $A1$, $B0$, $B1$, I, R) and a Boolean class Y defined by $Y = (A0 \wedge A1) \vee (B0 \wedge B1)$. Features $A0$, $A1$, $B0$, and $B1$ are independent to each other, feature I is uniformly random, and feature R matches the class Y 75% of the time. It is obvious that an optimal subset includes $A0$, $A1$, $B0$, and $B1$. I is irrelevant, and R is redundant. The other two data sets, Corral-47 and Corral-46, are obtained by introducing more irrelevant features and redundant features to the original Corral data. As its name says, Corral-47 contains a total of 47 Boolean features including 5 original features $A0$, $A1$, $B0$, $B1$, and R, 14 irrelevant features, and 28 additional redundant features. Among the 14 irrelevant features, only two features are uniformly random and each of the remaining 12 is completely correlated with either of the two. Among the 28 additional redundant features, for each of $A0$, $A1$, $B0$, and $B1$, there are 7 features that are correlated with it at various levels. The ratios of non-matches are $0, 1/16, 2/16, ..., 6/16$ respectively. Corral-46 is the same as Corral-47 except that it excludes R. Table 1 shows features selected by each algorithm. We use $A0$, $A1$, $B0$, $B1$ combined with subscripts $0, 1, ..., 6$ to represent the newly introduced redundant features, with the value of the subscripts indicating the ratio of non-matches.

We can see that for Corral, all the algorithms in comparison remove the irrelevant feature I, but fail to remove the redundant feature R. $\text{FCBF}_{(\log)}$ misses three features due to the setting of an improper threshold. For Corral-47, these algorithm also remove all the irrelevant features, but fail to remove R. The difference is that $\text{FCBF}_{(\log)}$, $\text{FCBF}_{(0)}$, and CFS-SF successfully remove all the additional redundant features. For Corral-46, only $\text{FCBF}_{(\log)}$ and $\text{FCBF}_{(0)}$ find the optimal subset. In Corral-47 and Corral-46, the threshold in $\text{FCBF}_{(\log)}$ does not affect the selection results. These results suggest that when feature redundancy can only be identified based on feature subsets (e.g., the redundancy of R is defined by the subset of $A0$, $A1$, $B0$, and $B1$), FCBF may not successfully remove redundant features. This is a hard problem for most heuristic search algorithms as well.

	$FCBF_{(log)}$	$FCBF_{(0)}$	ReliefF	CFS-SF	FOCUS-SF
Corral	**R** *A*0	**R** *A*0 *A*1 *B*0 *B*1	*A*0 *B*0 **R** *B*1 *A*1	*A*0 *A*1 *B*0 *B*1 **R**	**R**
Corral-47	**R** *A*0 *A*1 *B*0 *B*1	**R** *A*0 *A*1 *B*0 *B*1	**R** $\mathbf{B1_1}$ *A*0 $\mathbf{A0_0}$ *B*1 $\mathbf{B1_0}$ *B*0 $\mathbf{B0_0}$ $\mathbf{B0_2}$ *A*1 $\mathbf{A1_0}$	*A*0 *A*1 *B*0 *B*1 **R**	*A*0 *A*1 $\mathbf{A1_2}$ *B*0 *B*1 **R**
Corral-46	*A*0 *A*1 *B*0 *B*1	*A*0 *A*1 *B*0 *B*1	*A*0 $\mathbf{A0_0}$ $\mathbf{B1_1}$ *B*0 $\mathbf{B0_0}$ *B*1 $\mathbf{B1_0}$ $\mathbf{B0_2}$ *A*1 $\mathbf{A1_0}$	*A*0 $\mathbf{A0_3}$ $\mathbf{A0_4}$ *A*1 $\mathbf{A1_2}$ $\mathbf{A1_4}$ *B*0 $\mathbf{B0_0}$ *B*1 $\mathbf{B1_1}$	*A*0 *A*1 $\mathbf{A1_3}$ $\mathbf{A1_4}$ *B*0 *B*1

Table 1: Features selected by each feature selection algorithm on synthetic data.

However, in high-dimensional data which often contains a large portion of irrelevant and/or redundant features (as in Corral-47 and Corral-46), FCBF can effectively remove redundant features. We next further verify its effectiveness as well as efficiency compared to other algorithms through various real-world data of high dimensionality.

5.3 Results and Discussions on Benchmark Data

In various machine learning domains, there are two forms of high-dimensional data. Traditionally, the dimensionality is usually thought high if data contains tens or hundreds of features. In this form of data, the number of instances is normally much larger than the dimensionality. In new domains such as text categorization and genomic microarray analysis, the dimensionality is in the order of thousands or even higher, and often greatly exceeds the number of instances. Therefore, we evaluate our method in comparison with others on high-dimensional data of both forms.

5.3.1 UCI BENCHMARK DATA

Title	Features	Instances	Classes
Lung-cancer	56	32	3
Promoters	57	106	2
Splice	60	3190	3
USCensus90	67	9338	3
CoIL2000	85	5822	2
Chemical	150	936	3
Musk2	166	6598	2
Arrhythmia	279	452	16
Isolet	617	1560	26
Multi-features	649	2000	10

Table 2: Summary of **UCI** benchmark data sets.

All together 10 data sets in the traditional form are selected from the UCI Machine Learning Repository[1] and the UCI KDD Archive.[2] These data sets contain various numbers of features, instances, and classes, as shown in Table 2. For each data set, we first run all the feature selection algo-

1. http://www.ics.uci.edu/∼mlearn/MLRepository.html
2. http://kdd.ics.uci.edu

rithms in comparison, and obtain the running time and selected features for each algorithm. For data sets containing features with continuous values, we apply the MDL discretization method (Fayyad and Irani, 1993) before applying FCBF, CFS-SF, and FOCUS-SF. For ReliefF, we use 5 neighbors and 30 instances throughout the experiments as suggested by Robnik-Sikonja and Kononenko (2003). We then apply NBC and C4.5 on both the original data set and each of the newly obtained data sets (with only selected features), and obtain overall accuracy of 10-fold cross-validation. All experiments were conducted on a Pentium IV PC with 1 GB RAM.

Table 3 records the running time for each feature selection algorithm. We can observe that $\text{FCBF}_{(0)}$ is consistently faster than CFS-SF and FOCUS-SF. The time savings from $\text{FCBF}_{(0)}$ become more obvious when the data dimensionality increases. In many cases especially compared with FOCUS-SF, the time savings are in degrees of magnitude. These results verify the superior computational efficiency of sequential forward selection with elimination applied by FCBF over greedy sequential search applied by CFS-SF and FOCUS-SF. Comparison between $\text{FCBF}_{(0)}$ and ReliefF shows that ReliefF is unexpectedly slow even though its time complexity is linear to dimensionality. The reason lies in that searching for nearest neighbors in ReliefF involves distance calculation which is more costly than the calculation of symmetrical uncertainty. When we compare $\text{FCBF}_{(0)}$ with $\text{FCBF}_{(\log)}$, it is clear that the setting of a larger relevance threshold γ further speeds up FCBF.

Title	$\text{FCBF}_{(\log)}$	$\text{FCBF}_{(0)}$	ReliefF	CFS-SF	FOCUS-SF
Lung-cancer	0.001	0.02	0.09	0.05	0.08
Promoters	0.001	0.02	0.06	0.03	0.16
Splice	0.20	0.55	0.89	0.55	16.59
USCensus90	0.30	0.50	2.94	0.52	77.67
CoIL2000	0.25	0.50	4.25	1.98	143.94
Chemical	0.05	0.05	1.36	0.28	6.56
Musk2	0.53	0.88	9.55	4.84	85.78
Arrhythmia	0.06	0.08	1.19	0.78	13.70
Isolet	0.42	3.05	10.05	93.94	107.33
Multi-Features	1.19	19.42	11.42	71.00	67.56

Table 3: Running time (seconds) for each feature selection algorithm on **UCI** data.

Table 4 records the number of features selected by each feature selection algorithm. We can see that all these algorithms achieve significant reduction of dimensionality by selecting only a small portion of the original features. $\text{FCBF}_{(\log)}$ on average selects the smallest number of features.

Tables 5 and 6 show the 10-fold cross-validation accuracy of NBC and C4.5 respectively. For each data set, we conduct Student's paired two-tailed t-Test in order to evaluate the statistical significance of the difference between two averaged accuracy values: one resulted from $\text{FCBF}_{(\log)}$ and the other resulted from one of $\text{FCBF}_{(0)}$, the full set, ReliefF, CFS-SF, and FOCUS-SF. Each value in a *p*-val column records the probability associated with the t-Test. The smaller the value, the more significant the difference of the two average values is. The last row (L/W/T) in each table summarizes over all data sets the losses/wins/ties in accuracy (at significance level 0.1) comparing various feature sets with those selected by $\text{FCBF}_{(\log)}$. We can see that in general $\text{FCBF}_{(\log)}$ achieves similar accuracy as $\text{FCBF}_{(0)}$. Therefore, the effectiveness of FCBF can be verified from the following two general trends: (1) $\text{FCBF}_{(\log)}$ improves or maintains the accuracy of both NBC and C4.5, and the

Title	$FCBF_{(log)}$	$FCBF_{(0)}$	ReliefF	CFS-SF	FOCUS-SF
Lung-cancer	4	6	5	8	4
Promoters	6	6	4	4	4
Splice	9	22	11	6	10
USCensus90	3	4	2	1	13
CoIL2000	3	5	12	10	29
Chemical	4	5	7	7	11
Musk2	2	2	2	10	11
Arrhythmia	5	12	25	25	24
Isolet	5	32	23	137	11
Multi-Features	27	130	14	87	7
Average	7	22	11	30	12

Table 4: Number of features selected by each feature selection algorithm on **UCI** data.

	$FCBF_{(log)}$	$FCBF_{(0)}$		Full Set		ReliefF		CFS-SF		FOCUS-SF	
Title	Acc	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val
Lung-cancer	83.33	86.67	0.34	78.33	0.34	84.17	0.85	86.67	0.34	87.5	0.46
Promoters	93.27	93.27	1	91.55	0.55	87.82	0.25	95.18	0.17	90.45	0.40
Splice	93.95	96.14	$\mathbf{0.00}^{+}$	95.52	$\mathbf{0.00}^{+}$	91.32	$\mathbf{0.00}^{-}$	93.54	0.24	94.36	$\mathbf{0.08}^{+}$
USCensus90	97.94	97.88	0.19	93.49	$\mathbf{0.00}^{-}$	97.97	0.17	97.99	0.65	97.87	0.44
CoIL2000	93.94	93.92	0.34	78.68	$\mathbf{0.00}^{-}$	93.89	0.66	92.92	$\mathbf{0.01}^{-}$	83.22	$\mathbf{0.00}^{-}$
Chemical	71.91	67.73	$\mathbf{0.02}^{-}$	60.90	$\mathbf{0.00}^{-}$	71.26	0.77	70.51	0.35	66.35	$\mathbf{0.00}^{-}$
Musk2	84.59	84.59	1	84.78	0.51	84.59	1	64.87	$\mathbf{0.00}^{-}$	83.53	$\mathbf{0.01}^{-}$
Arrhythmia	67.48	65.73	0.45	60.88	$\mathbf{0.01}^{-}$	55.79	$\mathbf{0.00}^{-}$	69.05	0.45	69.06	0.56
Isolet	50.06	83.33	$\mathbf{0.00}^{+}$	84.10	$\mathbf{0.00}^{+}$	60.90	$\mathbf{0.00}^{+}$	87.31	$\mathbf{0.00}^{+}$	71.03	$\mathbf{0.00}^{+}$
Multi-feat	95.9	95.65	0.50	94.1	$\mathbf{0.01}^{-}$	67.65	$\mathbf{0.00}^{-}$	96.15	0.64	93.7	$\mathbf{0.02}^{-}$
L/W/T	-	**1/2/7**		**5/2/3**		**3/1/6**		**2/1/7**		**4/2/4**	

Table 5: Accuracy of **NBC** on selected features for **UCI** data: Acc records 10-fold cross-validation accuracy rate (%) and *p*-Val records the probability associated with a paired two-tailed t-Test. The symbols "+" and "−" respectively identify statistically significant (at 0.1 level) wins or losses over $FCBF_{(log)}$.

improvement is more pronounced for NBC; and (2) $FCBF_{(log)}$ can achieve similar or even higher accuracy compared with other algorithms.

5.3.2 NIPS BENCHMARK DATA

Three data sets with very high dimensionality but relatively few instances are selected from the NIPS 2003 feature selection benchmark data sets.[3] All these data sets contain two classes and a large number of artificially introduced random features in addition to real features. A summary of the data sets is given in Table 7. For each data set, we conduct experiments following the same procedure as that used in UCI data. The results are shown in Tables 8, 9, 10, and 11.

From Table 8, we observe similar trends as those from UCI data except that (1) for Dexter and Dorothea data, CFS-SF did not produce running time results (hence, neither selected features nor

3. http://clopinet.com/isabelle/Projects/NIPS2003/

Title	$FCBF_{(log)}$	$FCBF_{(0)}$		Full Set		ReliefF		CFS-SF		FOCUS-SF	
	Acc	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val
Lung-cancer	86.67	86.67	1	80.83	0.17	84.17	0.34	84.17	0.34	84.17	0.34
Promoters	80.18	80.18	1	78.09	0.42	82.36	0.55	80.18	1	81.36	0.67
Splice	94.01	94.14	0.64	93.98	0.89	90.53	**0.00**$^-$	93.39	**0.00**$^-$	93.79	0.11
USCensus90	98.12	98.12	1	98.19	0.39	98.12	1	97.99	**0.00**$^-$	98.21	0.11
CoIL2000	94.02	94.02	1	93.87	0.12	94.02	1	94.02	1	93.97	0.39
Chemical	95.41	95.41	1	94.13	**0.01**$^-$	95.94	0.14	95.94	0.14	95.31	0.86
Musk2	91.35	91.35	1	96.91	**0.00**$^+$	88.00	**0.00**$^-$	95.79	**0.00**$^+$	95.45	**0.00**$^+$
Arrhythmia	71.47	68.80	0.19	67.70	**0.04**$^-$	69.02	**0.07**$^-$	68.58	0.13	67.02	**0.04**$^-$
Isolet	49.17	75.77	**0.00**$^+$	79.87	**0.00**$^+$	59.10	**0.00**$^+$	81.35	**0.00**$^+$	68.84	**0.00**$^+$
Multi-feat	92.45	93.65	**0.04**$^+$	94.3	**0.01**$^+$	78.65	**0.00**$^-$	94.7	**0.00**$^+$	91.75	0.42
L/W/T	-	**0/2/8**		**2/3/5**		**4/1/5**		**2/3/5**		**1/2/7**	

Table 6: Accuracy of **C4.5** on selected features for **UCI** data: Acc records 10-fold cross-validation accuracy rate (%) and *p*-Val records the probability associated with a paired two-tailed t-Test. The symbols "+" and "−" respectively identify statistically significant (at 0.1 level) wins or losses over $FCBF_{(log)}$.

Title	Features			Instances		
	Total	Real	Random	Total	Class 1	Class 2
Arcene	10000	7000	3000	100	44	56
Dexter	20000	9947	10053	300	150	150
Dorothea	100000	50000	50000	800	78	722

Table 7: Summary of **NIPS** benchmark data sets.

accuracy results) because the program ran out of memory after a period of considerably long time due to its quadratic space complexity; and (2) FCBF achieves tremendous time savings for this group of data sets, for instance, roughly 1 minute by $FCBF_{(log)}$ versus 4.5 hours by FOCUS-SF in Dorothea data. Results in Table 9 show that all the algorithms in comparison can dramatically reduce the dimensionality for this group of data sets. In spite of its impressive efficiency and capability of dimensionality reduction, the effectiveness of FCBF can still be clearly revealed by the results in Tables 10 and 11. As we can see, FCBF either improves or maintains the accuracy of both NBC and C4.5 for all the three data sets. In addition, for each of the three data sets, the highest accuracy is achieved by applying NBC on the feature subset selected by FCBF.

Title	$FCBF_{(log)}$	$FCBF_{(0)}$	ReliefF	CFS-SF	FOCUS-SF
Arcene	0.42	0.75	9.16	1108.66	21.39
Dexter	1.80	2.63	45.43	N/A	928.78
Dorothea	68.95	393.80	349.27	N/A	16470.92

Table 8: Running time (seconds) for each feature selection algorithm on **NIPS** data.

Title	$FCBF_{(log)}$	$FCBF_0$	ReliefF	CFS-SF	FOCUS-SF
Arcene	24	39	45	50	4
Dexter	35	35	71	N/A	23
Dorothea	50	96	137	N/A	21

Table 9: Number of features selected by each feature selection algorithm on **NIPS** data.

	$FCBF_{(log)}$	$FCBF_{(0)}$		Full Set		ReliefF		CFS-SF		FOCUS-SF	
Title	Acc	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val
Arcene	91.0	93.0	0.34	69.0	$\mathbf{0.00}^-$	69.0	$\mathbf{0.02}^-$	92.0	0.68	59.0	$\mathbf{0.00}^-$
Dexter	90.0	90.0	1	88.0	0.26	73.00	$\mathbf{0.00}^-$	N/A	N/A	90.0	1
Dorothea	97.5	98.38	$\mathbf{0.01}^+$	90.25	$\mathbf{0.00}^-$	94.38	$\mathbf{0.00}^-$	N/A	N/A	95.25	$\mathbf{0.00}^-$

Table 10: Accuracy of **NBC** on selected features for **NIPS** data: Acc records 10-fold cross-validation accuracy rate (%) and *p*-Val records the probability associated with a paired two-tailed t-Test. The symbols "+" and "−" respectively identify statistically significant (at 0.1 level) wins or losses over $FCBF_{(log)}$.

5.4 Summary

From the previous empirical study, we can conclude that FCBF can efficiently achieve high degree of dimensionality reduction and enhance or maintain predictive accuracy with selected features. Its proven efficiency and effectiveness compared with other algorithms through various synthetic and benchmark data sets suggest that FCBF is practical for feature selection of high-dimensional data. It is worthy to emphasize that feature subsets selected by FCBF are decoupled from the choice of learning algorithms. In other words, FCBF does not directly aim to increase the accuracy of a particular learning algorithm as wrapper algorithms do. In order to achieve better accuracy within affordable time, a wrapper algorithm based on an intended learning algorithm can be applied to the significantly reduced subset obtained from FCBF.

In FCBF, there is one parameter, the relevance threshold γ. As consistently shown from the benchmark data, different settings of γ affect the speed of FCBF. The closer γ is set to 1, the faster FCBF is. As shown from Corral-47 and Corral-46 as well as many of the benchmark data sets which may contain a large number of irrelevant and/or redundant features, speeding up the algorithm by

	$FCBF_{(log)}$	$FCBF_{(0)}$		Full Set		ReliefF		CFS-SF		FOCUS-SF	
Title	Acc	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val	Acc	*p*-Val
Arcene	83.0	82.0	0.76	76.0	0.17	65.0	$\mathbf{0.04}^-$	79.0	0.40	75.0	0.26
Dexter	83.67	83.67	1	76.33	$\mathbf{0.02}^-$	76.67	$\mathbf{0.08}^-$	N/A	N/A	90.00	$\mathbf{0.01}^+$
Dorothea	92.88	93.0	0.76	90.38	$\mathbf{0.01}^-$	93.38	0.68	N/A	N/A	96.5	$\mathbf{0.00}^+$

Table 11: Accuracy of **C4.5** on selected features for **NIPS** data: Acc records 10-fold cross-validation accuracy rate (%) and *p*-Val records the probability associated with a paired two-tailed t-Test. The symbols "+" and "−" respectively identify statistically significant (at 0.1 level) wins or losses over $FCBF_{(log)}$.

setting γ to a reasonably large value does not sacrifice the goodness of the selected subsets. However, a close look at the accuracy of individual data sets in Tables 5 and 6 reveals that in certain cases (e.g., Isolet data), $\text{FCBF}_{(\log)}$ results in significantly reduced accuracy than $\text{FCBF}_{(0)}$ due to the setting of overly high threshold. Therefore, when we do not have prior knowledge about data, an easy and safe way of applying FCBF is to set γ to the default value 0.

6. Conclusions

In this paper, we have identified the need for explicit redundancy analysis in feature selection, provided a formal definition of feature redundancy, and investigated the relationship between feature relevance and redundancy. We have proposed a new framework of efficient feature selection via relevance and redundancy analysis, and a correlation-based method which uses *C*-correlation for relevance analysis and both *C*- and *F*-correlations for redundancy analysis. A new feature selection algorithm FCBF is implemented and evaluated through extensive experiments comparing with three representative feature selection algorithms. The feature selection results are further verified by two different learning algorithms. Our method demonstrates its efficiency and effectiveness for feature selection in supervised learning in domains where data contains many irrelevant and/or redundant features.

Some future works are planed along the following directions. First, since symmetrical uncertainty measure only handles nominal or discrete values, our current method requires continuous values be discretized, which opens the opportunity to investigate how different discretization methods affect the performance of FCBF. Second, it would be interesting to explore measures that can handle all types of values or ways of combining different measures under our framework of relevance and redundancy analysis. Another direction is to investigate how our method can be extended to deal with regression problems in which the class contains continuous values. Moreover, additional effort is needed to experiment our method on genomic microarray data for informative gene selection and investigate how small samples affect the performance of feature selection.

Acknowledgments

We gratefully thank the Action Editor and anonymous reviewers for their constructive comments. This work is in part supported by the National Science Foundation (NSF grant No. 0127815, 0231448) and Prop 301 (No. ECR A601) at ASU.

References

H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2):279–305, 1994.

D. A. Bell and H. Wang. A formalism for relevance and its application in feature subset selection. *Machine Learning*, 41(2):175–195, 2000.

A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

A. L. Blum and R. L. Rivest. Training a 3-node neural networks is NP-complete. *Neural Networks*, 5:117 – 127, 1992.

M. Dash, K. Choi, P. Scheuermann, and H. Liu. Feature selection for clustering – a filter solution. In *Proceedings of the Second International Conference on Data Mining*, pages 115–122, 2002.

M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis: An International Journal*, 1(3):131–156, 1997.

M. Dash and H. Liu. Consistency-based search in feature selection. *Artificial Intelligence*, 151(1-2): 155–176, 2003.

J. G. Dy, C. E. Brodley, A. C. Kak, L. S. Broderick, and A. M. Aisen. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):373–378, 2003.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 359–366, 2000.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

D. D. Jensen and P. R. Cohen. Multiple comparisions in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.

G. H. John, R. Kohavi, and K. Pfleger. Irrelevant feature and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994.

Y. Kim, W. Street, and F. Menczer. Feature selection for unsupervised learning via evolutionary search. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292, 1996.

W. Lee, S. J. Stolfo, and K. W. Mok. Adaptive intrusion detection: A data mining approach. *AI Review*, 14(6):533 – 567, 2000.

H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002a.

H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Boston: Kluwer Academic Publishers, 1998. ISBN 0-7923-8198-X.

H. Liu, H. Motoda, and L. Yu. Feature selection with selective sampling. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 395–402, 2002b.

H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 319–327, 1996.

A. Miller. *Subset Selection in Regression*. Chapman & Hall/CRC, 2 edition, 2002.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

P. Mitra, C. A. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, 2002.

K. S. Ng and H. Liu. Customer retention via data mining. *AI Review*, 14(6):569 – 590, 2000.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 1988.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

M. Robnik-Sikonja and I. Kononenko. Theoretical and empirical analysis of Relief and ReliefF. *Machine Learning*, 53:23–69, 2003.

D. L. Swets and J. J. Weng. Efficient content-based image retrieval using automatic feature selection. In *IEEE International Symposium on Computer Vision*, pages 85–90, 1995.

I. H. Witten and E. Frank. *Data Mining - Pracitcal Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann Publishers, 2000.

E. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 601–608, 2001.

Y. Yang and J. O. Pederson. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.

L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *Proceedings of the twentieth International Conference on Machine Learning*, pages 856–863, 2003.

L. Yu and H. Liu. Redundancy based feature selection for microarray data. In *Proceedings of the Tenth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 737–742, 2004.

Journal of Machine Learning Research 5 (2004) 1225–1251 Submitted 4/2004; Published 10/04

Statistical Analysis of Some Multi-Category Large Margin Classification Methods

Tong Zhang TZHANG@WATSON.IBM.COM
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA

Editor: Bernhard Schölkopf

Abstract

The purpose of this paper is to investigate statistical properties of risk minimization based multi-category classification methods. These methods can be considered as natural extensions of binary large margin classification. We establish conditions that guarantee the consistency of classifiers obtained in the risk minimization framework with respect to the classification error. Examples are provided for four specific forms of the general formulation, which extend a number of known methods. Using these examples, we show that some risk minimization formulations can also be used to obtain conditional probability estimates for the underlying problem. Such conditional probability information can be useful for statistical inferencing tasks beyond classification.

1. Motivation

Consider a binary classification problem where we want to predict label $y \in \{\pm 1\}$ based on observation x. One of the most significant achievements for binary classification in machine learning is the invention of large margin methods, which include support vector machines and boosting algorithms.

Based on a set of training samples $(X_1,Y_1),\ldots,(X_n,Y_n)$, a large margin binary classification algorithm produces a decision function $\hat{f}(\cdot)$ by minimizing an empirical loss function that is often a convex upper bound of the binary classification error function. Given $\hat{f}(\cdot)$, the binary decision rule is to predict $y = 1$ if $\hat{f}(x) \geq 0$, and to predict $y = -1$ otherwise (the decision rule at $\hat{f}(x) = 0$ is not important).

In the literature, the following form of large margin binary classification is often encountered: we minimize the empirical risk associated with a convex function ϕ in a pre-chosen function class C_n that may depend on the sample size:

$$\hat{f}(\cdot) = \arg\min_{f(\cdot)\in C_n} \frac{1}{n}\sum_{i=1}^{n} \phi(f(X_i)Y_i). \tag{1}$$

Originally such a scheme was regarded as a compromise to avoid computational difficulties associated with direct classification error minimization, which often leads to an NP-hard problem. Some recent works in the statistical literature argued that such methods could be used to obtain conditional probability estimates. For example, see Friedman et al. (2000), Lin (2002), Schapire and Singer (1999), Zhang (2004), Steinwart (2003) for related studies. This point of view allows people to show the consistency of various large margin methods: that is, in the large sample limit, the obtained classifiers achieve the optimal Bayes error rate. For example, see Bartlett et al. (2003), Jiang (2004), Lugosi and Vayatis (2004), Mannor et al. (2003), Steinwart (2002, 2004), Zhang

(2004). The consistency of a learning method is certainly a very desirable property, and one may argue that a good classification method should at least be consistent in the large sample limit.

Although statistical properties of binary classification algorithms based on the risk minimization formulation (1) are quite well-understood due to many recent works such as those mentioned above, there are much fewer studies on risk minimization based multi-category problems which generalizes the binary large margin method (1). The complexity of possible generalizations may be one reason. Another reason may be that one can always estimate the conditional probability for a multi-category problem using the binary classification formulation (1) for each category, and then pick the category with the highest estimated conditional probability (or score).[1]

It is still useful to understand whether there are more natural alternatives, and what risk minimization formulations that generalize (1) can be used to yield consistent classifiers in the large sample limit. An important step toward this direction has recently been taken by Lee et al. (2004), where the authors proposed a multi-category extension of the support vector machine that is infinite-sample Bayes consistent (Fisher consistent). The purpose of this paper is to generalize their study so as to include a much wider class of risk minimization formulations that can lead to consistent classifiers in the infinite-sample limit. Moreover, combined with some relatively simple generalization analysis for kernel methods, we are able to show that with appropriately chosen regularization conditions, classifiers obtained from certain formulations can approach the optimal Bayes error in the large sample limit.

Theoretical analysis of risk minimization based multi-category large margin methods have started to draw more attention recently. For example, in Desyatnikov and Meir (2003), learning bounds for some multi-category convex risk minimization methods were obtained, although the authors did not study possible choices of Bayes consistent formulations. A related study can be found in Liu and Shen (2004), but again only for special formulations.

Although this paper studies a number of multi-category classification methods, we shall not try to argue which one is better practically, or to compare different formulations experimentally. One reason is that some methods investigated in this paper were originally proposed by different researchers, who have much more practical experience with the corresponding algorithms. Due to the scope of this paper, it is simply impossible for us to include a comprehensive empirical study without overlooking some engineering tricks. Casual experimental comparisons can lead to misleading conclusions. Therefore in this paper we only focus on asymptotic theoretical analysis. Although our analysis provides useful statistical insights (especially asymptotically), the performance of a learning algorithm may also be affected by factors which we do not considered here, especially for small-sample problems. We shall refer the readers to Rifkin and Klautau (2004) for a recent experimental study on some multi-category classification algorithms, although the issue of which algorithm may have better practical performance (and under what circumstances) is far from resolved.

We organize the paper as follows. Section 2 introduces the multi-category classification problem, and a general risk minimization based approach. In Section 3, we give conditions that guarantee the infinite-sample consistency of the risk minimization formulation. In Section 4, examples of the general formulation, which extend some existing methods in the literature, will be presented. We shall study their properties such as the associated statistical models and conditions that en-

1. This approach is often called one-versus-all in machine learning. Another main approach is to encode a multi-category classification problem into binary classification sub-problems. The consistency of such encoding schemes cannot be analyzed in our framework, and we shall not discuss them.

sure the infinite-sample consistency (ISC) of the resulting risk minimization estimators. Section 5 contains a relatively simple generalization analysis (which is not necessarily tight) for kernel multi-categorization methods. Our purpose is to demonstrate that with appropriately chosen regularization conditions, classifiers obtained from ISC risk minimization formulations can approach the optimal Bayes classifier in the large sample limit. Concluding remarks will be presented in Section 6.

2. Multi-Category Classification

We consider the following K-category classification problem: given an input vector x, we would like to predict its corresponding label $y \in \{1,\ldots,K\}$. Let $p(x)$ be a predictor of y which is a function of x. In the machine learning framework, the quality of this predictor can be measured by a loss function $L(p(x),y)$, and the data (X,Y) are drawn from an unknown underlying distribution D.

Given a set of training samples $(X_1,Y_1),\ldots,(X_n,Y_n)$, randomly drawn from D, our goal is to find a predictor $\hat{p}(x)$ so that the expected true loss of $\hat{p}$ given below is as small as possible:

$$\mathbf{E}_{X,Y} L(\hat{p}(X),Y),$$

where we use $\mathbf{E}_{X,Y}$ to denote the expectation with respect to the true (but unknown) underlying distribution D.

The loss function $L(p,y)$ can be regarded as a $K \times K$ cost matrix. In this paper, we are mainly interested in the simple but also the most important case of $0-1$ classification loss: we have a loss of 0 for correct prediction, and loss of 1 for incorrect prediction. We consider a slightly more general family of cost matrices, where the classification errors for different classes are penalized differently:

$$L(p,y) = \begin{cases} 0 & \text{if } p = y \\ a_y & \text{if } p \neq y, \end{cases} \tag{2}$$

where $a_y > 0$ ($y = 1,\ldots,K$) are K pre-defined positive numbers. If we let $a_y = 1$ ($y = 1,\ldots,K$), then we have the standard classification error. The more general cost-sensitive classification error in (2) is useful for many applications. For example, in some medical diagnosis applications, classifying a patient with cancer to the *no-cancer* category is much worse than classifying a patient without cancer to the *possible cancer* category (since in the latter case, a more thorough test can be performed to produce a more definite diagnosis).

Let $p(X) \in \{1,\ldots,K\}$ be a classifier. Its classification error under (2) is given by

$$\ell(p(\cdot)) := \mathbf{E}_X \sum_{c=1, c\neq p(X)}^{K} a_c P(Y = c|X). \tag{3}$$

If we know the conditional density $P(Y = c|X)$, then the optimal classification rule with the minimum loss in (3), often referred to as the *Bayes rule*, is given by

$$p_*(X) = \max_{c\in\{1,2,\ldots,K\}} a_c P(Y = c|X). \tag{4}$$

In binary classification with 0-1 classification error, the class rule can be obtained using the sign of a real-valued decision function. This can be generalized to K class classification problem as

follows: we consider K decision functions $f_c(x)$ where $c = 1,\ldots,K$ and we predict the label y of x as

$$T(\mathbf{f}(x)) := \arg\max_{c\in\{1,\ldots,K\}} f_c(x), \tag{5}$$

where we denote by $\mathbf{f}(x)$ the vector function $\mathbf{f}(x) = [f_1(x),\ldots,f_K(x)]$. In the following, we use bold symbols such as $\mathbf{f}$ to denote vectors, and $\mathbf{f}_c$ to denote its c-th component. We also use $\mathbf{f}(\cdot)$ to denote vector functions. If two or more components of $\mathbf{f}$ achieve the same maximum value, then we may choose any of them as $T(\mathbf{f})$. In this framework, $\mathbf{f}_c(x)$ is often regarded as a scoring function for category c that is correlated with how likely x belongs to category c (compared with the remaining $k-1$ categories).

Note that only the relative strength of the component $\mathbf{f}_c$ compared with the alternatives $\mathbf{f}_k$ ($k \neq c$) is important. In particular, the decision rule given in (5) does not change when we add the same numerical quantity to each component of $\mathbf{f}(x)$. This allows us to impose one constraint on the vector $\mathbf{f}(x)$ which decreases the degree of freedom K of the K-component vector $\mathbf{f}(x)$ to $K-1$. For example, in the binary classification case, we can enforce $\mathbf{f}_1(x)+\mathbf{f}_2(x) = 0$, and hence $f(x)$ can be represented as $[\mathbf{f}_1(x),-\mathbf{f}_1(x)]$. The decision rule in (5), which compares $\mathbf{f}_1(x) \geq \mathbf{f}_2(x)$, is equivalent to $\mathbf{f}_1(x) \geq 0$. This leads to the binary classification rule mentioned in the introduction.

In the multi-category case, one may also interpret the possible constraint on the vector function $\mathbf{f}(\cdot)$, which reduces its degree of freedom from K to $K-1$, based on the following observation. In many cases, we seek $\mathbf{f}_c(x)$ as a function of $p(Y = c|x)$. Since we have a constraint $\sum_{c=1}^{K} p(Y = c|x) = 1$ (implying that the degree of freedom for $p(Y = c|x)$ is $K-1$), the degree of freedom for f is also $K-1$ (instead of K). However, we shall point out that in the algorithms we formulate below, we may either enforce such a constraint that reduces the degree of freedom of f, or we do not impose any constraint, which keeps the degree of freedom of f to be K. The advantage of the latter is that it allows the computation of each $\mathbf{f}_c(x)$ to be decoupled. It is thus much simpler both conceptually and numerically. Moreover, it directly handles multiple-label problems where we may assign each x to multiple labels of $y \in \{1,\ldots,K\}$. In this scenario, we do not have a constraint.

In this paper, we consider an empirical risk minimization method to solve a multi-category problem, which is of the following general form:

$$\hat{\mathbf{f}}(\cdot) = \arg\min_{\mathbf{f}(\cdot)\in C_n} \frac{1}{n}\sum_{i=1}^{n} \Psi_{Y_i}(\mathbf{f}(X_i)), \tag{6}$$

where $\mathbf{f}(\cdot)$ is a K-component vector function, and C_n is a vector function class. Each $\Psi_Y(\cdot) : R^K \to R$ (indexed by class label $Y \in \{1,\ldots,K\}$) is a real-valued function that takes a K-component vector as its parameter. As we shall see later, this method is a natural generalization of the binary classification method (1). Note that one may consider an even more general form with $\Psi_Y(\mathbf{f}(X))$ replaced by $\Psi_Y(\mathbf{f}(X),X)$, which we don't study in this paper.

The general formulation (6) covers many traditional and newly proposed multi-category classification methods. Examples will be given in Section 4. Some of them such as some multi-category extensions of support vector machines are directly motivated by margin maximization (in the separable case). In general, as we shall see in Section 4, the function $\Psi_Y(\mathbf{f})$ should be chosen such that it favors a vector predictor $\mathbf{f}$ with the component $\mathbf{f}_Y$ corresponding to the observed class label Y larger than the alternatives $\mathbf{f}_k$ for $k \neq Y$. In this sense, it encourages the correct classification rule in (5) by implicitly maximizes the difference of $\mathbf{f}_Y$ and the remaining components $\mathbf{f}_k$ ($k \neq Y$). One may

interpret this effect as soft margin-maximization, and hence one may consider learning algorithms based on (6) generally as multi-category large margin methods.

Given the estimator $\hat{\mathbf{f}}(\cdot)$ from (6), the classification rule is based on (5) or some variants which we shall discuss later. The main purpose of the paper is to investigate the following two issues:

- Consistency: whether the classification error $\ell(\hat{\mathbf{f}}(\cdot))$ converges to $\ell(p_*(\cdot))$ where $p_*(\cdot)$ is the Bayes rule defined in (4).
- Probability model: the relationship of $\hat{\mathbf{f}}(X)$ and the conditional probability vector $[P(Y = c|X)]_{c=1,\ldots,K}$.

3. Approximation Estimation Decomposition

From the standard learning theory, one can expect that with appropriately chosen C_n, the solution $\hat{\mathbf{f}}(\cdot)$ of (6) approximately minimizes the true Ψ risk $\mathbf{E}_{X,Y}\Psi_Y(\hat{\mathbf{f}}(X))$ with respect to the unknown underlying distribution D within the vector function class C_n. The true risk of a vector function $\mathbf{f}(\cdot)$ can be rewritten as

$$\mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X)) = \mathbf{E}_X W(\mathbf{P}(\cdot|X), \mathbf{f}(X)), \tag{7}$$

where $\mathbf{P}(\cdot|X) = [P(Y = 1|X), \ldots, P(Y = K|X)]$ is the conditional probability, and

$$W(\mathbf{q}, \mathbf{f}) := \sum_{c=1}^{K} \mathbf{q}_c \Psi_c(\mathbf{f}). \tag{8}$$

Note that we use $\mathbf{q}_c$ to denote the component c of a K-dimensional vector $\mathbf{q} \in \Lambda$, where Λ_K is the set of possible conditional probability vectors:

$$\Lambda_K := \left\{ \mathbf{q} \in R^K : \sum_{c=1}^{K} \mathbf{q}_c = 1,\ \mathbf{q}_c \geq 0 \right\}.$$

The vector argument $\mathbf{q}$ of $W(\mathbf{q}, \mathbf{f})$ represents the conditional probability vector evaluated at some point x; the argument $\mathbf{f}$ represents the value of our vector predictor evaluated at the same point x. Intuitively, $W(\mathbf{q}, \mathbf{f})$ is the point-wise true loss of $\mathbf{f}$ at some x, with respect to the conditional probability distribution $\mathbf{q} = [P(Y = \cdot|X = x)]$.

In order to understand the large sample behavior of the algorithm based on solving (6), we first need to understand the behavior of a vector function $\mathbf{f}(\cdot)$ that approximately minimizes $\mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X))$. We introduce the following definition. The property has also been referred to as *classification calibrated* in Bartlett et al. (2003) or *Fisher consistent* in Lin (2002). In this paper, we explicitly call it as *infinite-sample consistent*.

Definition 1 *Consider $[\Psi_c(\mathbf{f})]$ in (7). We say that the formulation is infinite-sample consistent (ISC) on a set $\Omega \subseteq R^K$ with respect to the classification error loss (3), if the following conditions hold:*

- *For each c, $\Psi_c(\cdot) : \Omega \to R$ is bounded below and continuous.*
- *$\forall \mathbf{q} \in \Lambda_K$ and $c \in \{1, \ldots, K\}$ such that $a_c \mathbf{q}_c < \sup_k a_k \mathbf{q}_k$, we have*

$$W^*(\mathbf{q}) := \inf_{\mathbf{f} \in \Omega} W(\mathbf{q}, \mathbf{f}) < \inf \left\{ W(\mathbf{q}, \mathbf{f}) : \mathbf{f} \in \Omega, \mathbf{f}_c = \sup_k \mathbf{f}_k \right\}.$$

Remark 2 *Among the two conditions, the second is more essential. It says that (point-wisely) for each conditional probability vector $\mathbf{q} \in \Lambda_K$, an exact optimal solution of $W(\mathbf{q},\cdot)$ leads to a Bayes rule with respect to the classification error defined in (3). That is, the exact minimization of (7) leads to the exact minimization of classification error. This condition is clearly necessary for consistency. The first condition (continuity) is needed to show that point-wisely, an approximate (instead of exact) minimizer of (7) also approximately minimizes the classification error.*

The following result relates the approximate minimization of the Ψ risk to the approximate minimization of classification error. The proof is left to Appendix B. A more general but also more abstract theory is presented in Appendix A.

Theorem 3 *Let $\mathcal{B}$ be the set of all vector Borel measurable functions (with respect to some underlying topology on the input space) which take values in R^K. For $\Omega \subset R^K$, let $\mathcal{B}_\Omega = \{\mathbf{f} \in \mathcal{B} : \forall x, \mathbf{f}(x) \in \Omega\}$. If $[\Psi_c(\cdot)]$ is ISC on Ω with respect to (3), then $\forall \varepsilon_1 > 0$, $\exists \varepsilon_2 > 0$ such that for all underlying Borel probability measurable D, and $\mathbf{f}(\cdot) \in \mathcal{B}_\Omega$,*

$$\mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X)) \leq \inf_{\mathbf{f}' \in \mathcal{B}_\Omega} \mathbf{E}_{X,y}\Psi_Y(\mathbf{f}'(X)) + \varepsilon_2$$

implies

$$\ell(T(\mathbf{f}(\cdot))) \leq \ell_B + \varepsilon_1,$$

$T(\cdot)$ is defined in (5), and ℓ_B is the optimal Bayes error: $\ell_B = \ell(p_*(\cdot))$, with p_* given in (4).

Based on the above theorem, an ISC risk minimization formulation is suitable for multi-category classification problems. The classifier obtained from minimizing (6) can approach the Bayes error rate if we can show that with appropriately chosen function class C_n, approximate minimization of (6) implies approximate minimization of (7). Learning bounds of this kind have been very well-studied in statistics and machine learning. For example, for binary classification, such bounds can be found in Blanchard et al. (2003), Bartlett et al. (2003), Jiang (2004), Lugosi and Vayatis (2004), Mannor et al. (2003), Steinwart (2002, 2004), Zhang (2004), where they were used to prove the consistency of various large margin classification methods. In order to achieve consistency, it is also necessary to take a sequence of function classes C_n (typically, one takes a sequence $C_1 \subset C_2 \subset \cdots \subset \mathcal{B}_\Omega$) such that $\cup_n C_n$ is dense (e.g. with respect to the uniform-norm topology) in $\mathcal{B}_\Omega$. This method, widely studied in the statistics literature, is often referred to as *the method of sieves* (for example, see Chapter 10 of van de Geer, 2000, and references therein). It is also closely related to the structural risk minimization method of Vapnik (1998). The set C_n has the effect of regularization, which ensures that for large n, $\mathbf{E}_{X,Y}\Psi_Y(\hat{\mathbf{f}}(X)) \approx \inf_{\mathbf{f}(\cdot)\in C_n} \mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X))$. It follows that as $n \to \infty$, $\mathbf{E}_{X,Y}\Psi_Y(\hat{\mathbf{f}}(X)) \xrightarrow{P} \inf_{\mathbf{f}(\cdot)\in\mathcal{B}_\Omega} \mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X))$. Theorem 3 then implies that $\ell(T(\hat{\mathbf{f}}(\cdot))) \xrightarrow{P} \ell_B$. The above idea, although intuitively clear, is not rigorously stated at this point. A rigorous treatment can be found in Section 5.

We can see that there are two types of errors in this framework. The first type of error, often referred to as *approximation error*, measures how close we are from the optimal Bayes error when we approximately minimize the true risk with respect to the surrogate loss function Ψ in (7). Theorem 3 implies that the approximation error goes to zero when we approximately minimize (7). The second type of error, often referred to as *estimation error*, is how close we are from achieving the minimum of the true Ψ risk in (7), when we obtain a classifier based on the empirical minimization

of (6). The overall statistical error of the risk minimization based classification method (6) is given by the combination of approximation error and estimation error.

Before studying learning bounds that relate approximate minimization of (6) to the approximate minimization of (7), we provide examples of Ψ that lead to ISC formulations. We pay special attention to the case that each $\Psi_c(\mathbf{f})$ is a convex function of $\mathbf{f}$, so that the resulting formulation becomes computationally more tractable (assuming we also use convex function classes C_n).

4. Multi-Category Classification Formulations

We give some examples of ISC multi-category classification formulations. They are motivated from methods proposed in the literature, and will be extended in our framework.

The following simple result says that an ISC formulation for an arbitrary loss of the form (2) can be obtained from an ISC formulation of any particular loss in that family.

Proposition 4 *Assume $[\Psi_c(\mathbf{f})]$ is ISC on $\Omega \subset R^K$ with respect to (3) with $a_c = a'_c$ ($c = 1,\ldots,K$). Then $\forall$ positive numbers a''_c ($c = 1,\ldots,K$), $[\Psi_c(\mathbf{f})a''_c/a'_c]$ is ISC on $\Omega \subset R^K$ with respect to (3) with $a_c = a''_c$ ($c = 1,\ldots,K$).*

Proof The first condition of ISC holds automatically. Now we shall check the second condition. For all $\mathbf{q} \in \Lambda_K$, we define $\mathbf{q}'$ as $\mathbf{q}'_c = \mathbf{q}_c a''_c/a'_c$. Therefore

$$\sum_{c=1}^{K} \mathbf{q}_c \frac{\Psi_c(\mathbf{f})a''_c}{a'_c} = \sum_{c=1}^{K} \mathbf{q}'_c \Psi_c(\mathbf{f}).$$

The ISC condition of $[\Psi_c(\mathbf{f})]$ with respect to $\{a'_c\}$ implies

$$\inf\left\{\sum_{c=1}^{K} \mathbf{q}_c \frac{\Psi_c(\mathbf{f})a''_c}{a'_c} : \mathbf{f} \in \Omega, \mathbf{f}_c = \sup_k \mathbf{f}_k\right\} > \inf_{\mathbf{f}\in\Omega} \sum_{c=1}^{K} \mathbf{q}_c \frac{\Psi_c(\mathbf{f})a''_c}{a'_c}$$

for all c such that $a'_c\mathbf{q}'_c < \sup_k a'_k\mathbf{q}'_k$. That is, for all c such that $a''_c\mathbf{q}_c < \sup_k a''_k\mathbf{q}_k$. This gives the second condition of ISC. ■

Due to the above result, for notational simplicity, we shall focus on the 0-1 classification error in this section, with $a_c = 1$ in (3):

$$\ell(p(\cdot)) = \mathbf{E}_X \sum_{c=1, c\neq p(X)}^{K} P(Y = c|X) = 1 - \mathbf{E}_X P(Y = p(\cdot)|X). \tag{9}$$

4.1 Pairwise Comparison Method

This model is motivated from the multi-class support vector machine in Weston and Watkins (1998).[2] Here we consider a more general formulation with the following choice of Ψ:

$$\Psi_c(\mathbf{f}) = \sum_{k=1}^{K} \phi(\mathbf{f}_c - \mathbf{f}_k), \tag{10}$$

2. According to Schölkopf and Smola. (2002), page 213, an identical method was proposed independently by Blanz et al. (1995) three years earlier in a talk given at AT&T.

where ϕ is an appropriately chosen real-valued function. The choice in Weston and Watkins (1998) is the hinge loss for the SVM formulation: $\phi(p) = (1-p)_+$.

Typically we choose a decreasing function ϕ in (10). Assume that we observe a datum X with its label Y. The intuition behind (10) is to favor a large value $\mathbf{f}_Y(X) - \mathbf{f}_k(X)$ for $k \neq Y$, which encourages the correct classification rule. This approach has some attractive features. Since it makes pairwise comparisons, the penalty term $\phi(\mathbf{f}_c - \mathbf{f}_k)$ can be adjusted in a pairwise fashion. This can be useful for some cost-sensitive classification problems that are more general than the particular form we consider in (3). With a differentiable ϕ (thus excludes the SVM hinge loss), this method also has the very desirable property of *order preserving*, which we state below.

Theorem 5 *Consider the formulation in (10). Let $\phi(\cdot) : R \to R$ be a non-increasing function such that $\phi(z) < \phi(-z)$ for all $z > 0$. Consider any $\mathbf{q} \in \Lambda_K$ and $\mathbf{f}$ such that $W(\mathbf{q},\mathbf{f}) = W^*(\mathbf{q})$. If $\mathbf{q}_i < \mathbf{q}_j$, we have $\mathbf{f}_i \leq \mathbf{f}_j$. Moreover, if $\phi(\cdot)$ is differentiable and $\phi'(0) < 0$, then we have $\mathbf{f}_i < \mathbf{f}_j$.*

Proof We can take $i = 1$ and $j = 2$. Let $\mathbf{f}' = \mathbf{f}_k$ when $k > 2$, $\mathbf{f}'_1 = \mathbf{f}_2$, and $\mathbf{f}'_2 = \mathbf{f}_1$. We now prove the first part by contradiction. Assume $\mathbf{f}_1 > \mathbf{f}_2$. We have

$$\begin{aligned}
&W(\mathbf{q},\mathbf{f}') - W(\mathbf{q},\mathbf{f}) \\
=&(\mathbf{q}_2 - \mathbf{q}_1)\left[\phi(\mathbf{f}_1 - \mathbf{f}_2) - \phi(\mathbf{f}_2 - \mathbf{f}_1) + \sum_{k>2}(\phi(\mathbf{f}_1 - \mathbf{f}_k) - \phi(\mathbf{f}_2 - \mathbf{f}_k))\right] \\
<&(\mathbf{q}_2 - \mathbf{q}_1)[0+0] = 0.
\end{aligned}$$

This is a contradiction to the optimality of $\mathbf{f}$. Therefore we must have $\mathbf{f}_1 \leq \mathbf{f}_2$, which proves the first part.

Now we assume in addition that $\phi(\cdot)$ is differentiable. Then at the optimal solution, we have the first order condition $\frac{\partial}{\partial \mathbf{f}_c} W(\mathbf{q},\mathbf{f}) = 0$:

$$\mathbf{q}_c \sum_{k=1}^{K} \phi'(\mathbf{f}_c - \mathbf{f}_k) = \sum_{k=1}^{K} \mathbf{q}_k \phi'(\mathbf{f}_k - \mathbf{f}_c).$$

Again, we prove the second part by contradiction. To this end let us assume $\mathbf{f}_1 = \mathbf{f}_2 = f$, then the above equality implies that

$$\mathbf{q}_1 \sum_{k=1}^{K} \phi'(f - \mathbf{f}_k) = \mathbf{q}_2 \sum_{k=1}^{K} \phi'(f - \mathbf{f}_k).$$

This is not possible since $\sum_{k=1}^{K} \phi'(f - \mathbf{f}_k) \leq 2\phi'(0) < 0$. ■

Note that for functions that are not differentiable, even if $\mathbf{q}_1 < \mathbf{q}_2$, we may still allow $\mathbf{f}_1 = \mathbf{f}_2$ at an optimal solution. Moreover, it is possible that the formulation is not ISC. We provide such a counter-example for the hinge loss in Appendix C. However, for differentiable functions, the method is infinite-sample consistent.

Theorem 6 *Let $\phi(\cdot) : R \to R$ be a differentiable non-negative and non-increasing function such that $\phi'(0) < 0$. Then the formulation (10) is ISC on $\Omega = R^K$ with respect to (9).*

Proof Consider $\mathbf{q} \in \Lambda_K$, and assume that $\mathbf{q}_1 < \mathbf{q}_2$. We show that

$$\inf\{W(\mathbf{q},\mathbf{f}) : \mathbf{f} \in \Omega, \mathbf{f}_1 \geq \mathbf{f}_2\} > W^*(\mathbf{q}).$$

This will imply ISC. We again prove by contradiction. If the claim is not true, then we can find sequences $\mathbf{f}^{(m)}$ such that $0 = \mathbf{f}_1^{(m)} \geq \mathbf{f}_2^{(m)}$ and $\lim_m W(\mathbf{q},\mathbf{f}^{(m)}) = W^*(\mathbf{q})$. We can further select subsequences such that for each pair i and j, $\mathbf{f}_i^{(m)} - \mathbf{f}_j^{(m)}$ converges (may converge to $\pm\infty$). This gives a limiting vector $\mathbf{f}$, with properly defined $\mathbf{f}_i - \mathbf{f}_j$ even when either $\mathbf{f}_i$ or $\mathbf{f}_j$ is $\pm\infty$. It follows from the assumption that $W(\mathbf{q},\mathbf{f}) = W^*(\mathbf{q})$ and $0 = \mathbf{f}_1 \geq \mathbf{f}_2$. However, this violates Theorem 5 (with trivial modification of the proof to handle the infinity-case), which asserts that $\mathbf{f}_1 < \mathbf{f}_2$. ■

A method closely related to (10) is to employ the following choice of Ψ (see Crammer and Singer, 2001):

$$\Psi_c(\mathbf{f}) = \phi(\mathbf{f}_c - \sup_{k \neq c} \mathbf{f}_k). \tag{11}$$

However, for convex ϕ, this method is usually not infinite-sample consistent. To see this, we assume that ϕ is a convex decreasing function and $\mathbf{q}_1 \geq \mathbf{q}_2 \cdots \geq \mathbf{q}_K$. After some simple algebra, we may choose $\mathbf{f}_1 \geq \mathbf{f}_2 = \cdots = \mathbf{f}_K$, and the corresponding $W(\mathbf{q},\mathbf{f}) = \mathbf{q}_1\phi(\mathbf{f}_1 - \mathbf{f}_2) - \sum_{k=2}^K \mathbf{q}_k\phi(\mathbf{f}_2 - \mathbf{f}_1)$. This means that unless $\mathbf{q}_1 > 0.5$, we can choose $\mathbf{f}_1 = \mathbf{f}_2$ to achieve the optimal value.

It is also worth mentioning that the formulation in (11) has been applied successfully in many practical applications. This may not be surprising since in many practical problems, the most important scenario is when the true label can be predicted relatively accurately. In such case (more precisely, when $\sup_k \mathbf{q}_k > 0.5$), the method is well behaved (ISC). The same reason is also why one may often successfully use (10) with the SVM hinge loss in practical problems, although from Appendix C, we know that the resulting classification method can be inconsistent. However, the analysis given in this section is still useful for the purpose of understanding the limitations of these methods.

4.2 Constrained Comparison Method

As pointed out, one may impose constraints on possible choices of $\mathbf{f}$. In this section, we consider another direct extension of binary large-margin method (1) to multi-category case. The choice given below is motivated by Lee et al. (2004), where an extension of SVM was proposed. For simplicity, we will consider linear equality constraint only:

$$\Psi_c(\mathbf{f}) = \sum_{k=1,k\neq c}^{K} \phi(-\mathbf{f}_k), \qquad \text{s.t.} \quad \mathbf{f} \in \Omega, \tag{12}$$

where we define Ω as

$$\Omega = \left\{ \mathbf{f} \in R^K : \sum_{k=1}^K \mathbf{f}_k = 0 \right\}.$$

Similar to the pairwise comparison model, if we choose a decreasing function ϕ in (10), then this formulation also encourages the correct classification rule. If we observe a datum X with its label Y, then the formulation favors small $\mathbf{f}_k(X)$ for all $k \neq Y$. Due to the sum to zero constraint, this implies a large $\mathbf{f}_Y(X)$.

We may interpret the added constraint in (12) as a restriction on the function class C_n in (6) such that every $\mathbf{f} \in C_n$ satisfies the constraint. Note that with $K = 2$, this leads to the standard binary large margin method.

Using (12), the conditional true Ψ risk (8) can be written as

$$W(\mathbf{q}, \mathbf{f}) = \sum_{c=1}^{K} (1 - \mathbf{q}_c)\phi(-\mathbf{f}_c), \quad \text{s.t. } \mathbf{f} \in \Omega. \tag{13}$$

Similar to the pairwise comparison model, for certain choices of function ϕ, this formulation has the desirable order preserving property.

Theorem 7 *Consider the formulation in (12), and assume that ϕ is strictly decreasing. Consider any $\mathbf{q} \in \Lambda_K$ and $\mathbf{f} \in \Omega$ such that $W(\mathbf{q}, \mathbf{f}) = W^*(\mathbf{q})$. If $\mathbf{q}_i < \mathbf{q}_j$, we have $\mathbf{f}_i \le \mathbf{f}_j$. Moreover, if ϕ is strictly convex and differentiable, then $\mathbf{f}_i < \mathbf{f}_j$.*

Proof The proof is rather straight forward. Let $i = 1$ and $j = 2$. Also let $\mathbf{f}'_k = \mathbf{f}_k$ when $k > 2$, $\mathbf{f}'_1 = \mathbf{f}_2$, and $\mathbf{f}'_2 = \mathbf{f}_1$. From $W(\mathbf{q}, \mathbf{f}') \ge W(\mathbf{q}, \mathbf{f})$, we obtain $(\mathbf{q}_1 - \mathbf{q}_2)(\phi(-\mathbf{f}_1) - \phi(-\mathbf{f}_2)) \ge 0$. This implies that $\phi(-\mathbf{f}_2) \ge \phi(-\mathbf{f}_1)$. Therefore $\mathbf{f}_1 \le \mathbf{f}_2$.

If ϕ is also differentiable, then using the Lagrangian multiplier method for the constraint $\sum_{c=1}^{K} \mathbf{f}_c = 0$, and differentiate at the optimal solution, we have $(1 - \mathbf{q}_1)\phi'(-\mathbf{f}_1) = (1 - \mathbf{q}_2)\phi'(-\mathbf{f}_2) = \lambda < 0$, where λ is the Lagrangian multiplier. The assumption $1 - \mathbf{q}_1 > 1 - \mathbf{q}_2$ implies that $\phi'(-\mathbf{f}_1) > \phi'(-\mathbf{f}_2)$. The strict convexity implies that $\mathbf{f}_1 < \mathbf{f}_2$. ■

The following result provides a simple way to check the infinite-sample consistency of (12). Note that since it only requires the differentiability on $(-\infty, 0]$, the SVM hinge loss is included.

Theorem 8 *If ϕ is a convex function which is bounded below, differentiable on $(-\infty, 0]$, and $\phi'(0) < 0$, then (12) is infinite-sample consistency on Ω with respect to (9).*

Proof The continuity condition is straight-forward to verify. We may also assume that $\phi(\cdot) \ge 0$ without loss of generality.

Consider $\mathbf{q} \in \Lambda_K$. Without loss of generality, we can assume that $\mathbf{q}_1 < \mathbf{q}_2$, and only need to show that $\inf\{W(\mathbf{q}, \mathbf{f}) : \mathbf{f} \in \Omega, \mathbf{f}_1 = \sup_k \mathbf{f}_k\} > W^*(\mathbf{q})$. Now consider a sequence $\mathbf{f}^{(m)}$ such that $\lim_m W(\mathbf{q}, \mathbf{f}^{(m)}) = \inf\{W(\mathbf{q}, \mathbf{f}) : \mathbf{f} \in \Omega, \mathbf{f}_1 = \sup_k \mathbf{f}_k\}$. Note that $(1 - \mathbf{q}_1)\phi(-\mathbf{f}_1^{(m)})$ is bounded.

Now if the sequence $\{\mathbf{f}^{(m)}\}$ is unbounded, then due to the constraint $\sum_k \mathbf{f}_k^{(m)} = 0$ and $\mathbf{f}_1^{(m)} \ge \mathbf{f}_k^{(m)}$, we know that the sequence $\{\mathbf{f}_1^{(m)}\}$ must also be unbounded. It follows that there is a subsequence (which for simplicity, denote as the whole sequence) such that $\mathbf{f}_1^{(m)} \to +\infty$. The boundedness of $(1 - \mathbf{q}_1)\phi(-\mathbf{f}_1^{(m)})$ implies that $\mathbf{q}_1 = 1$, which is not possible since $\mathbf{q}_1 < \mathbf{q}_2$.

Therefore we know that the sequence $\{\mathbf{f}^{(m)}\}$ must be bounded, and thus it contains a convergent subsequence. Denote the limit as $\mathbf{f}$. We have $W(\mathbf{q}, \mathbf{f}) = \lim_m W(\mathbf{q}, \mathbf{f}^{(m)})$. Therefore we only need to show that $W(\mathbf{q}, \mathbf{f}) > W^*(\mathbf{q})$. We consider three cases:

- $\mathbf{f}_1 = \mathbf{f}_2$. Since $\mathbf{f}_1 = \sup_k \mathbf{f}_k$, we have $\mathbf{f}_1 = \mathbf{f}_2 \ge 0$. The convexity assumption implies that $\phi'(-\mathbf{f}_1) = \phi'(-\mathbf{f}'_2) \le \phi'(0) < 0$. Therefore $(1 - \mathbf{q}_1)\phi'(-\mathbf{f}_1) - (1 - \mathbf{q}_2)\phi'(-\mathbf{f}_2) < 0$. It follows that there is a sufficiently small δ such that $(1 - \mathbf{q}_1)\phi(-\mathbf{f}_1 + \delta) + (1 - \mathbf{q}_2)\phi(-\mathbf{f}_2 - \delta) < (1 - \mathbf{q}_1)\phi(-\mathbf{f}_1) + (1 - \mathbf{q}_2)\phi(-\mathbf{f}_2)$. Therefore if we let $\mathbf{f}'_1 = \mathbf{f}_1 - \delta$, $\mathbf{f}'_2 = \mathbf{f}_2 + \delta$, and $\mathbf{f}'_k = \mathbf{f}_k$ when $k > 2$, then $W(\mathbf{q}, \mathbf{f}) > W(\mathbf{q}, \mathbf{f}') \ge W^*(\mathbf{q})$.

- $\mathbf{f}_1 > \mathbf{f}_2$ and $\phi(-\mathbf{f}_1) > \phi(-\mathbf{f}_2)$. In this case, if we let $\mathbf{f}'_1 = \mathbf{f}_2$, $\mathbf{f}'_2 = \mathbf{f}_1$, and $\mathbf{f}'_k = \mathbf{f}_k$ when $k > 2$, then it is easy to check that $W(\mathbf{q},\mathbf{f}) - W^*(\mathbf{q}) \leq W(\mathbf{q},\mathbf{f}) - W(\mathbf{q},\mathbf{f}') = (\mathbf{q}_1 - \mathbf{q}_2)(\phi(-\mathbf{f}_2) - \phi(-\mathbf{f}_1)) > 0$.

- $\mathbf{f}_1 > \mathbf{f}_2$ and $\phi(-\mathbf{f}_1) \leq \phi(-\mathbf{f}_2)$. Using the condition that $-\mathbf{f}_1 < 0$ and hence $\phi'(-\mathbf{f}_1) \leq \phi'(0) < 0$, we know that for a sufficiently small $\delta > 0$, we have $\phi(-\mathbf{f}_1 + \delta) < \phi(-\mathbf{f}_1) \leq \phi(-\mathbf{f}_2)$ and $-\mathbf{f}_2 - \delta > -\mathbf{f}_1$. Since the convexity of ϕ implies that $\phi(z)$ achieves the maximum on $[-\mathbf{f}_1, -\mathbf{f}_2]$ at its end points, we have $\phi(-\mathbf{f}_2) \geq \phi(-\mathbf{f}_2 - \delta)$. Therefore if we let $\mathbf{f}'_1 = \mathbf{f}_1 - \delta$, $\mathbf{f}'_2 = \mathbf{f}_2 + \delta$, and $\mathbf{f}'_k = \mathbf{f}_k$ when $k > 2$, then $W(\mathbf{q},\mathbf{f}) > W(\mathbf{q},\mathbf{f}') \geq W^*(\mathbf{q})$.

Combining the above three cases, we obtain the result. ■

Using the above criterion, we can convert an ISC convex ϕ for the binary formulation (1) into an ISC multi-category classification formulation (12). In Lee et al. (2004) the special case of SVM (with loss function $\phi(z) = (1-z)_+$ which is convex and differentiable on $(-\infty, 0]$) was studied. The authors demonstrated the infinite-sample consistency by direct calculation, although no results similar to Theorem 3, needed for proving consistency, were established. The treatment presented here generalizes their study.

4.3 One-Versus-All Method

The constrained comparison method in (12) is closely related to the one-versus-all approach, where we use the formulation (1) to train one function $\mathbf{f}_c(X)$ for each class c separately but regarding all data (X,Y) such that $Y \neq c$ as negative data, and all data (X,Y) such that $Y = c$ as positive data. It can be easily checked that the resulting formulation is a special case of (6) with

$$\Psi_c(\mathbf{f}) = \phi(\mathbf{f}_c) + \sum_{k=1,k\neq c}^{K} \phi(-\mathbf{f}_k). \tag{14}$$

Note that this formula is similar to (12), but we don't require the sum-of-zero constraint on $\mathbf{f}$ (that is $\Omega = R^K$). Intuitively, with an observation (X,Y), this formulation encourages the correct classification rule in that it favors a large $\mathbf{f}_Y(X)$ and favors small $\mathbf{f}_k(X)$ when $k \neq Y$. However, if a binary classification method (such as SVM) does not estimate the conditional probability, then the one-versus-all approach may not be infinite-sample consistent, while the formulation in (12) can still be. In order to establish the ISC condition for the one-versus-all approach, we can write

$$W(\mathbf{q},\mathbf{f}) = \sum_{c=1}^{K} \left[\mathbf{q}_c \phi(\mathbf{f}_c) + (1-\mathbf{q}_c)\phi(-\mathbf{f}_c)\right]. \tag{15}$$

We have the following order-preserving property.

Theorem 9 *Consider (14). Assume that ϕ is convex, bounded below, differentiable, and $\phi(z) < \phi(-z)$ when $z > 0$. Consider any $\mathbf{q} \in \Lambda_K$ and $\mathbf{f} \in [-\infty, +\infty]^K$ such that $W(\mathbf{q},\mathbf{f}) = W^*(\mathbf{q})$. If $\mathbf{q}_i < \mathbf{q}_j$, we have $\mathbf{f}_i < \mathbf{f}_j$.*

Proof Let f_q (not necessarily unique) minimizes $q\phi(f) + (1-q)\phi(-f)$. We have the first-order optimality condition

$$q\phi'(f_q) = (1-q)\phi'(-f_q).$$

Note that the assumptions imply that $\phi'(0) < 0$. Therefore $f_q \neq 0$ when $q \neq 0.5$ (otherwise, the optimality condition cannot be satisfied). Therefore by the assumption that $\phi(z) < \phi(-z)$ when $z > 0$, we have $f_q > 0$ when $q > 0.5$ and $f_q < 0$ when $q < 0.5$.

Let $i = 1$ and $j = 2$. We have either $\mathbf{q}_1 \in [0, 0.5)$ or $\mathbf{q}_2 \in (0.5, 1]$. Assume the former (due to the symmetry, the latter case can be proved similarly), which implies that $\mathbf{f}_1 < 0$. If $\mathbf{f}_2 \geq 0$, then the claim $\mathbf{f}_1 < \mathbf{f}_2$ holds. Therefore we only need to consider the case $\mathbf{f}_2 < 0$, and thus $0 \leq \mathbf{q}_1 < \mathbf{q}_2 \leq 0.5$. We now prove by contradiction. Note that $\mathbf{f}_2 > -\infty$ (otherwise, $\mathbf{q}_2\phi(\mathbf{f}_2) = +\infty$). If $\mathbf{f}_2 \leq \mathbf{f}_1 < 0$, then the convexity of ϕ implies $\phi'(\mathbf{f}_2) \leq \phi'(\mathbf{f}_1) < 0$. We have

$$\phi'(-\mathbf{f}_1) = \mathbf{q}_1\phi'(\mathbf{f}_1)/(1-\mathbf{q}_1) > \mathbf{q}_2\phi'(\mathbf{f}_1)/(1-\mathbf{q}_2) \geq \mathbf{q}_2\phi'(\mathbf{f}_2)/(1-\mathbf{q}_2) = \phi'(-\mathbf{f}_2).$$

The convexity implies that $-\mathbf{f}_1 > -\mathbf{f}_2$ (thus $\mathbf{f}_1 < \mathbf{f}_2$), which is a contradiction. Therefore we must have $\mathbf{f}_1 < \mathbf{f}_2$. ■

The following result shows that for a (non-flat) differentiable convex function ϕ, the one-versus-all method is infinite-sample consistent. Note that the theorem excludes the standard SVM method, which employs the non-differentiable hinge loss. However, similar to the discussion at the end of Section 4.1, if the true label can be predicted relatively accurately (that is, the dominant class has a conditional probability larger than 0.5), then the SVM one-versus-all method is consistent. Therefore the method may still perform well for some practical problems (see Rifkin and Klautau, 2004, for example).

Theorem 10 *Under the assumptions of Theorem 9. The method (14) is ISC on $\Omega = R^K$ with respect to (9).*

Proof Consider $\mathbf{q} \in \Lambda_K$. Without loss of generality, we can assume that $\mathbf{q}_1 < \mathbf{q}_2$, and only need to show that $\inf\{W(\mathbf{q},\mathbf{f}) : \mathbf{f} \in \Omega, \mathbf{f}_1 = \sup_k \mathbf{f}_k\} > W^*(\mathbf{q})$. Now consider a sequence $\mathbf{f}^{(m)}$ such that $\lim_m W(\mathbf{q},\mathbf{f}^{(m)}) = \inf\{W(\mathbf{q},\mathbf{f}) : \mathbf{f} \in \Omega, \mathbf{f}_1 = \sup_k \mathbf{f}_k\}$. Let $\mathbf{f}$ be a limiting point of $\mathbf{f}^{(m)}$ in $[-\infty,+\infty]^K$, we have $W(\mathbf{q},\mathbf{f}) = \lim_m W(\mathbf{q},\mathbf{f}^{(m)})$ and $\mathbf{f}_1 = \sup_k \mathbf{f}_k$. From Theorem 9, we have $W(\mathbf{q},\mathbf{f}) > W^*(\mathbf{q})$. ■

Using Theorem 24, we can also obtain a more quantitative bound.

Theorem 11 *Under the assumptions of Theorem 9. The function $V_\phi(q) = \inf_{f \in R}[q\phi(f) + (1-q)\phi(-f)]$ is concave on $[0,1]$. Assume that there exists a constant $c_\phi > 0$ such that*

$$(q-q')^2 \leq c_\phi^2 \left(2V_\phi(\frac{q+q'}{2}) - V_\phi(q) - V_\phi(q') \right),$$

then we have $\forall \mathbf{f}(\cdot)$,

$$\ell(T(\mathbf{f}(\cdot))) \leq \ell_B + c_\phi \left(\mathbf{E}_{X,Y}\Phi_Y(\mathbf{f}(X)) - \inf_{\mathbf{f}'} \mathbf{E}_{X,Y}\Phi_Y(\mathbf{f}'(X)) \right)^{1/2},$$

where $\Phi_Y(\mathbf{f})$ is given in (14), $T(\cdot)$ is defined in (5), ℓ is the 0-1 classification error in (9), and ℓ_B is the optimal Bayes error.

Proof $V_\phi(q)$ is the infimum of concave functions $q\phi(f)+(1-q)\phi(-f)$ indexed by $f \in R$, thus concave.

The second part is an application of Theorem 24. We use the notations of Appendix A: let $\mathcal{X}$ be the input space, $\mathcal{Q} = \Lambda_K$ be the space of conditional probability vectors, and $\mathcal{D} = \{1,\ldots,K\}$ be the space of class labels. We let $\ell(\mathbf{q},k) = \sum_{c=1,c\neq k} \mathbf{q}_c$, and thus the classification error of a decision function $p(\cdot)$ in (9) can be expressed as $\ell(p(\cdot)) = \mathbf{E}_X \ell([P(Y=c|X)]_c, p(X))$. The estimation-model space is R^K, with decision T given by (5). The W function is given by (15). Let $v(\mathbf{q}) \equiv 1$. $\forall \varepsilon > 0$, assume $\Delta\ell(\mathbf{q}, T(\mathbf{f})) \geq \varepsilon$.

Define $V_\phi(q,f) = q\phi(f)+(1-q)\phi(-f)$. Without loss of generality, we may assume that $T(\mathbf{f}) = 1$ and $\mathbf{q}_2 = \sup_c \mathbf{q}_c$. Then $\Delta\ell(\mathbf{q}, T(\mathbf{f})) = \mathbf{q}_2 - \mathbf{q}_1 \geq \varepsilon$.

$$\begin{aligned}
\Delta W(\mathbf{q},\mathbf{f}) &\geq \inf_{\mathbf{f}_1 \geq \mathbf{f}_2} \sum_{i=1}^{2} \left[V_\phi(\mathbf{q}_i,\mathbf{f}_i) - V_\phi(\mathbf{q}_i)\right] \\
&= \inf_{\mathbf{f}_1 = \mathbf{f}_2} \sum_{i=1}^{2} \left[V_\phi(\mathbf{q}_i,\mathbf{f}_i) - V_\phi(\mathbf{q}_i)\right] \\
&= 2 \inf_{\mathbf{f}_1} V_\phi\left(\frac{\mathbf{q}_1+\mathbf{q}_2}{2}, \mathbf{f}_1\right) - (V_\phi(\mathbf{q}_1)+V_\phi(\mathbf{q}_2)) \geq c_\phi^{-2}(\mathbf{q}_1-\mathbf{q}_2)^2 \geq c_\phi^{-2}\varepsilon^2.
\end{aligned}$$

The first equality holds because the minimum cannot be achieved at a point $\mathbf{f}_1 < \mathbf{f}_2$ due to the order-preserving property in Theorem 9. The assumption thus implies that $c_\phi^2 \Delta H_{\ell,W,T,v}(\varepsilon) \geq \varepsilon^2$. The desired result is now a direct consequence of Theorem 24. ■

Remark 12 *Using Taylor expansion, it is easy to verify that the condition $V_\phi''(q) \leq -c < 0$ implies that $(2V_\phi((q+q')/2) - V_\phi(q) - V_\phi(q')) \geq c(q-q')^2/4$. In this case, we may take $c_\phi = 2/\sqrt{c}$. As an example, we consider the least squares method and one of its variants: $\phi(z) = (1-v)^2$ or $\phi(z) = (1-v)_+^2$. In both cases, $V_\phi(q) = 4q(1-q)$. Therefore we can let $c_\phi = 1/\sqrt{2}$.*

The bound can also be further refined under the so-called Tsybakov small noise assumption (see Mammen and Tsybakov, 1999).

Theorem 13 *Under the assumptions of Theorem 11. Let*

$$\gamma(X) = \inf\{\sup_c P(Y=c|X) - P(Y=c'|X) : P(Y=c'|X) < \sup_c P(Y=c|X)\}$$

be the margin between the largest conditional probability and the second largest conditional probability (let $\gamma(X) = 1$ if all conditional probabilities are equal). Consider $\alpha \geq 0$ such that $c_\gamma = \mathbf{E}_X \gamma(X)^{-\alpha} < +\infty$, then we have $\forall \mathbf{f}(\cdot)$,

$$\ell(T(\mathbf{f}(\cdot))) \leq \ell_B + c_\phi^{(2\alpha+2)/(\alpha+2)} \left(\mathbf{E}_{X,Y}\Phi_Y(\mathbf{f}(X)) - \inf_{\mathbf{f}'(\cdot)} \mathbf{E}_{X,Y}\Phi_Y(\mathbf{f}'(X))\right)^{(\alpha+1)/(\alpha+2)} c_\gamma^{1/(\alpha+2)}.$$

Proof Using notations in the proof of Theorem 11, but let $v(\mathbf{q}) = \gamma(\mathbf{q})^{-\alpha}/c_\gamma$, where $\gamma(\mathbf{q}) = \inf\{\sup_c \mathbf{q}_c - \mathbf{q}_k : \mathbf{q}_k < \sup_c \mathbf{q}_c\}$. It is clear that $\mathbf{E}_X v(\mathbf{q}(X)) = 1$ with $\mathbf{q}(X) = [P(Y=1|X), \cdots, P(Y=K|X)]$.

Following the proof of Theorem 11, but assume $\mathbf{q}_2 - \mathbf{q}_1 \geq \varepsilon\nu(\mathbf{q})$. From $\mathbf{q}_2 - \mathbf{q}_1 \geq \gamma(\mathbf{q})$, we have $\forall \beta \geq 0$: $(\mathbf{q}_2 - \mathbf{q}_1)^{1+\beta}/\gamma(\mathbf{q})^{-\alpha+\beta} \geq (\mathbf{q}_2 - \mathbf{q}_1)/\gamma(\mathbf{q})^{-\alpha} \geq \varepsilon/c_\gamma$. Let $\beta = \alpha/(\alpha+2)$, we have

$$\left((\mathbf{q}_2 - \mathbf{q}_1)^2/\gamma(\mathbf{q})^{-\alpha}\right)^{(\alpha+1)/(\alpha+2)} \geq \varepsilon/c_\gamma.$$

This implies that (the first inequality follows from the proof of Theorem 11)

$$\Delta W(\mathbf{q},\mathbf{f})/\nu(\mathbf{q}) \geq c_\phi^{-2}(\mathbf{q}_1 - \mathbf{q}_2)^2/\nu(\mathbf{q}) \geq \varepsilon^{(\alpha+2)/(\alpha+1)} c_\gamma^{-1/(\alpha+1)} c_\phi^{-2}.$$

Thus $c_\gamma^{1/(\alpha+1)} c_\phi^2 \Delta H_{\ell,W,T,\nu}(\varepsilon) \geq \varepsilon^{(\alpha+2)/(\alpha+1)}$. The bound now follows directly from Theorem 24. ■

4.4 Unconstrained Background Discriminative Method

We consider the following unconstrained formulation:

$$\Psi_c(\mathbf{f}) = \psi(\mathbf{f}_c) + s\left(\sum_{k=1}^K t(\mathbf{f}_k)\right), \tag{16}$$

where ψ, s and t are appropriately chosen convex functions that are continuously differentiable. As we shall see later, this is a generalization of the maximum-likelihood method, which corresponds to $s(z) = t(z) = 1$ and $\psi(z) = -\ln(z)$.

We shall choose s and t such that the unconstrained background term $s\left(\sum_{k=1}^K t(\mathbf{f}_k)\right)$ penalizes large $\mathbf{f}_k$ for all k. We also choose a decreasing $\psi(\mathbf{f}_c)$ so that it favors a large $\mathbf{f}_c$. That is, it serves the purpose of discriminating $\mathbf{f}_c$ against the background term. The overall effect is to favor a predictor in which $\mathbf{f}_c$ is larger than $\mathbf{f}_k$ ($k \neq c$). In (16), the first term has a relatively simple form that depends only on the label c. The second term is independent of the label, and can be regarded as a normalization term. Note that this function is symmetric with respect to components of $\mathbf{f}$. This choice treats all potential classes equally. It is also possible to treat different classes differently. For example, replacing $\psi(\mathbf{f}_c)$ by $\psi_c(\mathbf{f}_c)$ or replacing $t(\mathbf{f}_k)$ by $t_k(\mathbf{f}_k)$.

4.4.1 Optimality Equation and Probability Model

Using (16), the conditional true Ψ risk (8) can be written as

$$W(\mathbf{q},\mathbf{f}) = \sum_{c=1}^K \mathbf{q}_c \psi(\mathbf{f}_c) + s\left(\sum_{c=1}^K t(\mathbf{f}_c)\right).$$

In the following, we study the property of the optimal vector $\mathbf{f}^*$ that minimizes $W(\mathbf{q},\mathbf{f})$ for a fixed $\mathbf{q}$.

Given $\mathbf{q}$, the optimal solution $\mathbf{f}^*$ that minimizes $W(\mathbf{q},\mathbf{f})$ satisfies the following first order optimality condition:

$$\mathbf{q}_c \psi'(\mathbf{f}_c^*) + \mu_{\mathbf{f}^*} t'(\mathbf{f}_c^*) = 0 \qquad (c = 1,\ldots,K). \tag{17}$$

where the quantity $\mu_{\mathbf{f}^*} = s'(\sum_{k=1}^K t(\mathbf{f}_k^*))$ is independent of c.

Clearly this equation relates $\mathbf{q}_c$ to $\mathbf{f}_c^*$ for each component c. The relationship of $\mathbf{q}$ and $\mathbf{f}^*$ defined by (17) can be regarded as the (infinity sample-size) probability model associated with the learning method (6) with Ψ given by (16). The following result is quite straight-forward. We shall skip the proof.

Theorem 14 *Assume that ψ, t, s are differentiable functions such that $s'(x) > 0$. If for $a \in [0, +\infty)$, the the solution x of $a\psi'(x) + t'(x) = 0$ is an increasing function of a, then the solution of (17) has the order preserving property: $\mathbf{q}_i < \mathbf{q}_j$ implies $\mathbf{f}_i^* < \mathbf{f}_j^*$. Moreover, the method (16) is ISC.*

In the following, we shall present various formulations of (16) which have the order preserving property.

4.4.2 DECOUPLED FORMULATIONS

We let $s(u) = u$ in (16). The optimality condition (17) becomes

$$\mathbf{q}_c \psi'(\mathbf{f}_c^*) + t'(\mathbf{f}_c^*) = 0 \qquad (c = 1, \ldots, K). \tag{18}$$

This means that we have K decoupled equalities, one for each $\mathbf{f}_c$. This is the simplest and in the author's opinion, the most interesting formulation. Since the estimation problem in (6) is also decoupled into K separate equations, one for each component of $\hat{\mathbf{f}}$, this class of methods are computationally relatively simple and easy to parallelize. Although this method seems to be preferable for multi-category problems, it is not the most efficient way for two-class problems (if we want to treat the two classes in a symmetric manner) since we have to solve two separate equations. We only need to deal with one equation in (1) due to the fact that an effective constraint $\mathbf{f}_1 + \mathbf{f}_2 = 0$ can be used to reduce the number of equations. This variable elimination has little impact if there are many categories.

In the following, we list some examples of multi-category risk minimization formulations. They all have the order preserving property, hence are infinite-sample consistent. We focus on the relationship of the optimal optimizer function $\mathbf{f}_*(\mathbf{q})$ and the conditional probability $\mathbf{q}$, which gives the probability model.

$\psi(u) = -u$ AND $t(u) = e^u$

We obtain the following probability model: $\mathbf{q}_c = e^{\mathbf{f}_c^*}$. This formulation is closely related to the maximum-likelihood estimate with conditional model $\mathbf{q}_c = e^{\mathbf{f}_c^*} / \sum_{k=1}^K e^{\mathbf{f}_k^*}$ (logistic regression). In particular, if we choose a function class such that the normalization condition $\sum_{k=1}^K e^{\mathbf{f}_k} = 1$ holds, then the two formulations are identical. However, they become different when we do not impose such a normalization condition.

$\phi(u) = -\ln u$ AND $t(u) = u$

This formulation is closely related to the previous formulation. It is an extension of maximum-likelihood estimate with probability model $\mathbf{q}_c = \mathbf{f}_c^*$. The resulting method is identical to the maximum-likelihood method if we choose our function class such that $\sum_k \mathbf{f}_k = 1$ and $\mathbf{f}_k \geq 0$ for $k = 1, \ldots, K$. However, the formulation also allows us to use function classes that do not satisfy the normalization constraint $\sum_k \mathbf{f}_k = 1$. Therefore this method is more flexible.

$\phi(u) = -\frac{1}{\alpha} u^\alpha$ $(0 < \alpha < 1)$ AND $t(u) = u$

Closely related to the maximum-likelihood method, this formulation replaces $\phi(u) = -\ln(u)$ by $\phi(u) = -u^\alpha$. The solution is $\mathbf{q}_c = (\mathbf{f}_c^*)^{1/(1-\alpha)}$. Similar to the case of $\phi(u) = -\ln(u)$, we may also impose a constraint $\sum_k \mathbf{f}_k^{1/(1-\alpha)} = 1$, which ensures that the estimated probability always sum to one.

$\phi(u) = -u$ AND $t(u) = \ln(1+e^u)$

This version uses binary logistic regression loss, and we have the following probability model: $\mathbf{q}_c = (1+e^{-\mathbf{f}_c^*})^{-1}$. Again this is an unnormalized model.

$\phi(u) = -u$ AND $t(u) = \frac{1}{p}|u|^p$ $(p > 1)$

We obtain the following probability model: $\mathbf{q}_c = \text{sign}(\mathbf{f}_c^*)|\mathbf{f}_c^*|^{p-1}$. This means that at the solution, $\mathbf{f}_c^* \geq 0$. This formulation is not normalized. If we choose a function family such that $\sum_k |\mathbf{f}_k|^{p-1} = 1$ and $\mathbf{f}_k \geq 0$, then we have a normalized model for which the estimated conditional probability always sum to one. One can also modify this method such that we can use $\mathbf{f}_c^* \leq 0$ to model the condition probability $\mathbf{q}_c = 0$.

$\phi(u) = -u$ AND $t(u) = \frac{1}{p}\max(u,0)^p$ $(p > 1)$

In this probability model, we have the following relationship: $\mathbf{q}_c = \max(\mathbf{f}_c^*, 0)^{p-1}$. The equation implies that we allow $\mathbf{f}_c^* \leq 0$ to model the conditional probability $\mathbf{q}_c = 0$. Therefore, with a fixed function class, this model is more powerful than the previous one. However, at the optimal solution, we still require that $\mathbf{f}_c^* \leq 1$. This restriction can be further alleviated with the following modification.

$\phi(u) = -u$ AND $t(u) = \frac{1}{p}\min(\max(u,0)^p, p(u-1)+1)$ $(p > 1)$

In this model, we have the following relationship at the solution: $\mathbf{q}_c = \min(\max(\mathbf{f}_c^*, 0), 1)^{p-1}$. Clearly this model is more powerful than the previous model since the function value $\mathbf{f}_c^* \geq 1$ can be used to model $\mathbf{q}_c = 1$. For separable problems, at each point there exists a c such that $\mathbf{q}_c = 1$ and $\mathbf{q}_k = 0$ when $k \neq c$. The model requires that $\mathbf{f}_c^* \geq 1$ and $\mathbf{f}_k^* \leq 0$ when $k \neq c$. This is essentially a large margin separation condition, where the function for the true class is separated from the rest by a margin of one.

4.4.3 COUPLED FORMULATIONS

In the coupled formulation with $s(u) \neq u$, the probability model are inherently normalized in some sense. We shall just list a few examples.

$\phi(u) = -u$, AND $t(u) = e^u$, AND $s(u) = \ln(u)$

This is the standard logistic regression model. The probability model is

$$\mathbf{q}_c(x) = \frac{e^{\mathbf{f}_c^*(x)}}{\sum_{c=1}^K e^{\mathbf{f}_c^*(x)}}.$$

The right hand side is always normalized (sum up to 1). One potential disadvantage of this method (at this moment, we don't know whether or not this theoretical disadvantage causes real problems in practice or not) is that it does not model separable data very well. That is, if $\mathbf{q}_c(x) = 0$ or $\mathbf{q}_c(x) = 1$, we require $\mathbf{f}_c^* = \pm\infty$. In comparison, some large margin methods described earlier can model the separable scenario using finite valued $\mathbf{f}^*$.

$\phi(u) = -u$, AND $t(u) = |u|^{p'}$, AND $s(u) = \frac{1}{p}|u|^{p/p'}$ $(p, p' > 1)$

The probability model is

$$\mathbf{q}_c(x) = \left(\sum_{k=1}^{K} |\mathbf{f}_k^*(x)|^{p'}\right)^{(p-p')/p'} \text{sign}(\mathbf{f}_c^*(x))|\mathbf{f}_c^*(x)|^{p'-1}.$$

We may replace $t(u)$ by $t(u) = \max(0,u)^p$, and the probability model becomes

$$\mathbf{q}_c(x) = \left(\sum_{k=1}^{K} \max(\mathbf{f}_k^*(x),0)^{p'}\right)^{(p-p')/p'} \max(\mathbf{f}_c^*(x),0)^{p'-1}.$$

These formulations do not seem to have advantages over the decoupled counterparts (with $s(u) = 1$). For the decoupled counterparts, as explained, the normalization (so that the estimated probability sum to one) can be directly included into the function class. This is more difficult to achieve here due to the more complicated formulations. However, it is unclear whether normalized formulations have practical advantages since one can always explicitly normalize the estimated conditional probability.

5. Consistency of Kernel Multi-Category Classification Methods

In this section, we give conditions that lead to the consistency of kernel methods. It is worth mentioning that generalization bounds obtained in this section are not necessarily tight. We use simple analysis to demonstrate that statistical consistency can be obtained. In order to obtain good rate of convergence results, more sophisticated analysis (such as those used by Blanchard et al., 2004, Bartlett et al., 2003, Mannor et al., 2003, van de Geer, 2000, Scovel and Steinwart, 2003) is needed.

The analysis given in this section is kernel independent. Therefore we can start with an arbitrary reproducing kernel Hilbert space H (for example, see Wahba, 1990, for definition) with inner product $\cdot$ and norm $\|\cdot\|_H$. Each element of H is a function $f(x)$ of the input x. It is well known that for each data point x, we can embed it into H as h_x such that $f(x) = f \cdot h_x$ for all $f \in H$.

In this section, we only consider bounded input distribution D:

$$\sup_x \|h_x\|_H < \infty.$$

We also introduce the following notations:

$$H_A = \{f(\cdot) \in H : \|f\|_H \sup_x \|h_x\|_H \le A\},$$
$$H_{A,K} = H_A^K = \{\mathbf{f}(\cdot) : \mathbf{f}_c(\cdot) \in H_A \text{ for all } c = 1,\ldots,K\}.$$

For notation simplicity, we shall limit our discussion to formulations such that for all $c = 1,\ldots,K$, $\Psi_c(\cdot)$ defined on a subset $\Omega \subset R^K$ can be extended to R^K. For example, for the constrained comparison model with the SVM loss. we require that $\Omega = \{\mathbf{f} \in R^K : \sum_{k=1}^K \mathbf{f}_k = 0\}$, but the formulation itself is well-defined on the entire R^K.

In order to obtain a uniform convergence bound, we shall introduce the following Lipschitz condition. It is clear that all well-behaved formulations such as those considered in this paper satisfy this assumption.

Assumption 15 *Given any $A > 0$, and consider $S_A = \{\mathbf{f} \in R^K : \sup_c |\mathbf{f}_c| \leq A\}$. Then there exists γ_A such that $\forall \mathbf{f}, \mathbf{f}' \in S_A$ and $1 \leq c \leq K$:*

$$|\Psi_c(\mathbf{f}) - \Psi_c(\mathbf{f}')| \leq \gamma_A \sup_k |\mathbf{f}_k - \mathbf{f}'_k|.$$

Definition 16 *Let $Q_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ be a set of n points. We define the $\ell_\infty(Q_n)$ distance between any two functions $f(x,y)$ and $g(x,y)$ as*

$$\ell_\infty(Q_n)(f,g) = \sup_i |f(X_i, Y_i) - g(X_i, Y_i)|.$$

Let $\mathcal{F}$ be a class of functions of (x,y), the empirical ℓ_∞-covering number *of $\mathcal{F}$, denoted by $N(\varepsilon, \mathcal{F}, \ell_\infty(Q_n))$, is the minimal number of balls $\{g : \ell_\infty(Q_n)(g,f) \leq \varepsilon\}$ of radius ε needed to cover $\mathcal{F}$. The* uniform ℓ_∞ covering number *is given by*

$$N_\infty(\varepsilon, \mathcal{F}, n) = \sup_{Q_n} N(\varepsilon, F, \ell_\infty(Q_n)),$$

where the supremum is over all samples Q_n of size n.

Note that we may also use other covering numbers such as ℓ_2 covering numbers. The ℓ_∞ covering number is more suitable for the specific Lipschitz condition used in Assumption 15. We use the following kernel-independent covering number bound.

Lemma 17 *Consider the function class $\mathcal{F}_{A,K} = \{\Psi_Y(\mathbf{f}(X)) : \mathbf{f} \in H_{A,K}\}$ such that Ψ satisfies Assumption 15. Then there exists a universal constant $C_1 > 0$ such that*

$$\ln N_\infty(\gamma_A \varepsilon, \mathcal{F}_{A,K}, n) \leq K C_1 A^2 \frac{\ln(2 + A/\varepsilon) + \ln n}{n\varepsilon^2}.$$

Proof Note that Theorem 4 of Zhang (2002) implies that there exists C_1 such that

$$\ln N_\infty(\varepsilon, H_A, n) \leq C_1 A^2 \frac{\ln(2 + A/\varepsilon) + \ln n}{n\varepsilon^2}.$$

Therefore with empirical samples $Q_n = \{(X_i, Y_i)\}$, we can find $\exp(K C_1 A^2 \frac{\ln(2+A/\varepsilon)+\ln n}{n\varepsilon^2})$ vectors $\mathbf{f}^j(X_i)$ such that for each $\mathbf{f} \in H_{A,K}$, we have $\inf_j \sup_{i,c} |\mathbf{f}_c(X_i) - \mathbf{f}^j_c(X_i)| \leq \varepsilon$. The assumption implies that this is a cover of $\mathcal{F}_{A,K}$ of radius $\gamma_A \varepsilon$. ■

Remark 18 *For specific kernels, the bound can usually be improved. Moreover, the* log*-covering number (entropy) depends linearly on the number of classes K. This is due to the specific regularization condition we use here. For practical problems, it can be desirable to use other regularization conditions so that the corresponding covering numbers have much weaker dependency (or even independence) on K. For simplicity, we will not discuss such issues in this paper.*

Lemma 19 *Consider function class $\mathcal{F}_{A,K} = \{\Psi_Y(\mathbf{f}(X)) : \mathbf{f} \in H_{A,K}\}$ such that Ψ satisfies Assumption 15. Then there exists a universal constant C such that for all $n \geq 2$:*

$$\mathbf{E}_{Q_n} \sup_{\mathbf{f}\in\mathcal{F}_{A,K}} \left| \frac{1}{n}\sum_{i=1}^{n} \Psi_{Y_i}(\mathbf{f}(X_i)) - \mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X)) \right| \leq C\sqrt{K}\frac{\gamma_A A \ln^{3/2} n}{\sqrt{n}},$$

where $\mathbf{E}_{Q_n}$ denotes the expectation over empirical training data $Q_n = \{(X_i, Y_i)\}$.

Proof Let $\mathbf{f}_0 \in H_{A,K}$, and define $\mathcal{F}^0_{A,K} = \{\Psi_Y(\mathbf{f}(X)) - \Psi_Y(\mathbf{f}_0(X)) : \mathbf{f} \in \mathcal{F}_{A,K}\}$. Consider a sequence of binary random variables such that $\sigma_i = \pm 1$ with probability $1/2$. The *Rademacher complexity* of $\mathcal{F}^0_{A,K}$ under empirical sample $Q_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ is given by

$$R(\mathcal{F}^0_{A,K}, Q_n) = \mathbf{E}_\sigma \sup_{\mathbf{f}\in H_{A,K}} \left| \frac{1}{n}\sum_{i=1}^{n} \sigma_i(\Psi_{Y_i}(\mathbf{f}(X_i)) - \Psi_{Y_i}(\mathbf{f}_0(X_i))) \right|.$$

It is well known that there exists a universal constant C_2 (a variant of Corollary 2.2.8 in van der Vaart and Wellner, 1996):

$$R(\mathcal{F}^0_{A,K}, Q_n) \leq C_2 \inf_{\varepsilon_0} \left[\varepsilon_0 + \frac{1}{\sqrt{n}} \int_{\varepsilon_0}^{\infty} \sqrt{\log N_\infty(\varepsilon, \mathcal{F}, Q_n)}\, d\varepsilon \right].$$

Using the bound in Lemma 17, and perform the integration with $\varepsilon_0 = \gamma_A A\sqrt{1/n}$, we obtain

$$R(\mathcal{F}^0_{A,K}, Q_n) \leq \frac{C\sqrt{K}}{2}\frac{\gamma_A A \ln^{3/2} n}{\sqrt{n}},$$

where C is a universal constant.

Now using the standard symmetrization argument (for example, see Lemma 2.3.1 of van der Vaart and Wellner, 1996), we have

$$\mathbf{E}_{Q_n} \sup_{\mathbf{f}\in\mathcal{F}_{A,K}} \left| \frac{1}{n}\sum_{i=1}^{n} \Psi_{Y_i}(\mathbf{f}(X_i)) - \mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X)) \right| \leq 2\mathbf{E}_{Q_n} R(\mathcal{F}^0_{A,K}, Q_n) \leq C\sqrt{K}\frac{\gamma_A A \ln^{3/2} n}{\sqrt{n}}.$$

■

Theorem 20 *Consider Ψ that satisfies Assumption 15. Choose A_n such that $A_n \to \infty$ and $\gamma_{A_n} A_n \ln^{3/2} n/\sqrt{n} \to 0$. Let $C_n = H_{A_n,K} \cap \mathcal{B}_\Omega$ (see Theorem 3 for the definition of $\mathcal{B}_\Omega$), where $\Omega \subset R^K$ is a constraint set. Consider the estimator $\hat{\mathbf{f}}(\cdot)$ in (6). We have*

$$\lim_{n\to\infty} \mathbf{E}_{Q_n} \mathbf{E}_{X,Y}\Psi_Y(\hat{\mathbf{f}}(X)) = \inf_{\mathbf{f}\in H\cap\mathcal{B}_\Omega} \mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X)).$$

Proof Consider $\mathbf{f}^{(n)} \in C_n$ that minimizes $\mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}(X))$. Since $\sum_{i=1}^n \Psi_{Y_i}(\hat{\mathbf{f}}(X_i)) \leq \sum_{i=1}^n \Psi_{Y_i}(\mathbf{f}^{(n)}(X_i))$, we have from Lemma 19 that

$$\mathbf{E}_{Q_n} \mathbf{E}_{X,Y}\Psi_Y(\hat{\mathbf{f}}(X)) \leq \mathbf{E}_{X,Y}\Psi_Y(\mathbf{f}^{(n)}(X)) + 2C\sqrt{K}\frac{\gamma_{A_n} A_n \ln^{3/2} n}{\sqrt{n}}.$$

Therefore as $n \to \infty$,

$$\lim_n \mathbf{E}_{Q_n} \mathbf{E}_{X,Y} \Psi_Y(\hat{\mathbf{f}}(X)) \to \lim_n \mathbf{E}_{X,Y} \Psi_Y(\mathbf{f}^{(n)}(X)) = \inf_{\mathbf{f} \in H \cap \mathcal{B}_\Omega} \mathbf{E}_{X,Y} \Psi_Y(\mathbf{f}(X)).$$

■

The following consistency result is a straight-forward consequence of Theorem 20 and Theorem 3.

Corollary 21 *Under the conditions of Theorem 20. Assume that Ψ is ISC on Ω with respect to (3). If H is dense in $\mathcal{B}_\Omega$, that is,*

$$\inf_{\mathbf{f}(\cdot) \in H \cap \mathcal{B}_\Omega} \mathbf{E}_{X,Y} \Psi_Y(\mathbf{f}(X)) = \inf_{\mathbf{f}(\cdot) \in \mathcal{B}_\Omega} \mathbf{E}_{X,Y} \Psi_Y(\mathbf{f}(X)),$$

then

$$\lim_{n \to \infty} \mathbf{E}_{Q_n} \mathbf{E}_{X,Y} \Psi_Y(\hat{\mathbf{f}}(X)) = \inf_{\mathbf{f}(\cdot) \in \mathcal{B}_\Omega} \mathbf{E}_{X,Y} \Psi_Y(\mathbf{f}(X)).$$

This implies that the classification error $\ell(\hat{\mathbf{f}})$ converges to the optimal Bayes error in probability.

6. Conclusion

In this paper we investigated a general family of risk minimization based multi-category classification algorithms, which can be considered as natural extensions of binary large margin methods. We established infinite-sample consistency conditions that ensure the statistical consistency of the obtained classifiers in the infinite-sample limit. Several specific forms of the general risk minimization formulation were considered. We showed that some models can be used to estimate conditional class probabilities. As an implication of this work, we see that it is possible to obtain consistent conditional density estimators using various non-maximum likelihood estimation methods. One advantage of some proposed large margin methods is that they allow us to model zero conditional probability directly. Note that for the maximum-likelihood method, near-zero conditional probability may cause robustness problems (at least in theory) due to the unboundedness of the log-loss function. Moreover, combined with some relatively simple generalization analysis, we showed that given appropriately chosen regularization conditions in some reproducing kernel Hilbert spaces, classifiers obtained from some multi-category kernel methods can approach the optimal Bayes error in the large sample limit.

Appendix A. Relationship of True Loss Minimization and Surrogate Loss Minimization

We consider an abstract decision model. Consider input space $\mathcal{X}$, output-model space Q, decision space $\mathcal{D}$, and estimation-model space Ω.

Consider the following functions:

- True loss function: $\ell : Q \times \mathcal{D} \to R$. We also define the corresponding excess loss as

$$\Delta\ell(\mathbf{q}, d) = \ell(\mathbf{q}, d) - \inf_{d' \in \mathcal{D}} \ell(\mathbf{q}, d').$$

- Surrogate loss function: $W : \mathcal{Q} \times \Omega \to R$. We also define the corresponding excess surrogate loss as
$$\Delta W(\mathbf{q}, \mathbf{f}) = W(\mathbf{q}, \mathbf{f}) - \inf_{\mathbf{f}' \in \mathcal{D}} W(\mathbf{q}, \mathbf{f}').$$
- Decision-rule: $T : \Omega \to \mathcal{D}$.

For the multi-category classification problem studied in the main text, $\mathcal{X}$ is the input space, $\mathcal{Q} = \Lambda_K$ is the space of conditional probability vectors $[P(Y = c|\cdot)]_c$, $\mathcal{D} = \{1, \ldots, K\}$ is the space of class labels, and $\Omega \subset R^K$ is the set of possible vector predictors $\mathbf{f} \in R^K$, with T given by (5). The W function is given by (8). With classification error in (2), we let

$$\ell(\mathbf{q}, k) = \sum_{c=1, c \neq k}^{K} a_c \mathbf{q}_c.$$

Therefore the classification error of a decision function $p(\cdot)$ in (3) can be expressed as

$$\ell(p(\cdot)) = \mathbf{E}_X \ell([P(Y = c|X)]_c, p(X)).$$

Definition 22 *Consider function $v : \mathcal{Q} \to R^+$. $\forall \varepsilon \geq 0$, we define*

$$\Delta H_{\ell,W,T,v}(\varepsilon) = \inf \left\{ \frac{\Delta W(\mathbf{q}, \mathbf{f})}{v(\mathbf{q})} : \Delta \ell(\mathbf{q}, T(\mathbf{f})) \geq \varepsilon v(\mathbf{q}) \right\} \cup \{+\infty\}.$$

The definition is designed so that the following properties hold. They are simple re-interpretations of the definition.

Proposition 23 *We have:*

- $\Delta H_{\ell,W,T,v}(\varepsilon) \geq 0$.
- $\Delta H_{\ell,W,T,v}(0) = 0$.
- $\Delta H_{\ell,W,T,v}(\varepsilon)$ *is non-decreasing on* $[0, +\infty)$.
- $v(\mathbf{q}) \Delta H_{\ell,W,T,v}(\Delta \ell(\mathbf{q}, T(\mathbf{f}))/v(\mathbf{q})) \leq \Delta W(\mathbf{q}, \mathbf{f})$.

The importance of the above definition is based on the following theorem. It essentially gives a bound on the expected excessive true loss ℓ using the expected excessive surrogate loss W. The idea was used by Bartlett et al. (2003), Zhang (2004) to analyze binary classification problems.

Theorem 24 *Given any distribution on $\mathcal{X}$, and function $v : \mathcal{Q} \to R^+$ such that*

$$\mathbf{E}_X v(\mathbf{q}(X)) = 1.$$

Let $\zeta(\varepsilon)$ be a convex function on $[0, +\infty)$ such that $\zeta(\varepsilon) \leq \Delta H_{\ell,W,T,v}(\varepsilon)$. Then $\forall \mathbf{f} : \mathcal{X} \to \Omega$, we have

$$\zeta(\mathbf{E}_X \Delta \ell(\mathbf{q}(X), T(\mathbf{f}(X)))) \leq \mathbf{E}_X \Delta W(\mathbf{q}(X), \mathbf{f}(X)).$$

Proof Using Jensen's inequality, we have

$$\zeta(\mathbf{E}_X \Delta\ell(\mathbf{q}(X), T(\mathbf{f}(X)))) \leq \mathbf{E}_X v(\mathbf{q}(X)) \zeta\left(\frac{\Delta\ell(\mathbf{q}(X), T(\mathbf{f}(X)))}{v(\mathbf{q}(X))}\right).$$

Now using the assumption and Proposition 23, we can upper-bound the right hand side by $\mathbf{E}_X \Delta W(\mathbf{q}(X), \mathbf{f}(X))$. This proves the theorem. ■

The following proposition is based mostly on Bartlett et al. (2003). We include it here for completeness.

Proposition 25 *Let $\zeta_*(\varepsilon) = \sup_{a\geq 0, b}\{a\varepsilon + b : \forall z \geq 0, az + b \leq \Delta H_{\ell,W,T,v}(z)\}$, then ζ_* is a convex function. It has the following properties:*

- $\zeta_*(\varepsilon) \leq \Delta H_{\ell,W,T,v}(\varepsilon)$.
- $\zeta_*(\varepsilon)$ *is non-decreasing.*
- *For all convex function ζ such that $\zeta(\varepsilon) \leq \Delta H_{\ell,W,T,v}(\varepsilon)$, $\zeta(\varepsilon) \leq \zeta_*(\varepsilon)$.*
- *Assume that $\exists a > 0$ and $b \in R$ such that $a\varepsilon + b \leq \Delta H_{\ell,W,T,v}(\varepsilon)$, and $\forall \varepsilon > 0, \Delta H_{\ell,W,T,v}(\varepsilon) > 0$. Then $\forall \varepsilon > 0, \zeta_*(\varepsilon) > 0$.*

Proof We note that ζ_* is the point-wise supreme of convex functions, thus it is also convex. We now prove the four properties.

- The first property holds by definition.
- The second property follows from the fact that $\Delta H_{\ell,W,T,v}(z)$ is non-decreasing, and $a\varepsilon' + b > a\varepsilon + b$ when $\varepsilon' > \varepsilon$.
- Given a convex function ζ such that $\zeta(\varepsilon) \leq \Delta H_{\ell,W,T,v}(\varepsilon)$. At any ε, we can find a line $az + b \leq \zeta(z) \leq \Delta H_{\ell,W,T,v}(z)$ and $\zeta(\varepsilon) = a\varepsilon + b$. This implies that $\zeta(\varepsilon) \leq \zeta_*(\varepsilon)$.
- Consider $\varepsilon > 0$. Using the fact that when $z \geq \varepsilon/2$, $\Delta H_{\ell,W,T,v}(z) \geq \Delta H_{\ell,W,T,v}(\varepsilon/2) > 0$, and the assumption, we know that there exists $a_\varepsilon \in (0, a)$ such that $a_\varepsilon(z - \varepsilon/2) < \Delta H_{\ell,W,T,v}(z)$. Therefore $\zeta_*(\varepsilon) \geq a_\varepsilon(\varepsilon - \varepsilon/2) > 0$.

■

The following result shows that the approximate minimization of the expected surrogate loss $\mathbf{E}_X \Delta W$ implies the approximate minimization of the expected true loss $\mathbf{E}_X \Delta\ell$.

Corollary 26 *Consider function $v : Q \to R^+$. If the loss function $\ell(\mathbf{q}, d)/v(\mathbf{q})$ is bounded, and $\forall \varepsilon > 0$, $\Delta H_{\ell,W,T,v}(\varepsilon) > 0$, then there exists a concave function ξ on $[0, +\infty)$ that depends only on ℓ, W, T, and v, such that $\xi(0) = 0$ and $\lim_{\delta\to 0^+} \xi(\delta) = 0$. Moreover, for all distribution on $\mathcal{X}$ such that $\mathbf{E}_X v(\mathbf{q}(X)) = 1$, we have*

$$\mathbf{E}_X \Delta\ell(\mathbf{q}(X), T(\mathbf{f}(X))) \leq \xi(\mathbf{E}_X \Delta W(\mathbf{q}(X), \mathbf{f}(X))).$$

Proof Consider $\zeta_*(\varepsilon)$ in Proposition 25. Let $\xi(\delta) = \sup\{\varepsilon : \varepsilon \geq 0, \zeta_*(\varepsilon) \leq \delta\}$. Then $\zeta_*(\varepsilon) \leq \delta$ implies that $\varepsilon \leq \xi(\delta)$. Therefore the desired inequality follows from Theorem 24. Given $\delta_1, \delta_2 \geq 0$: from $\zeta_*(\frac{\xi(\delta_1)+\xi(\delta_2)}{2}) \leq \frac{\delta_1+\delta_2}{2}$, we know that $\frac{\xi(\delta_1)+\xi(\delta_2)}{2} \leq \xi(\frac{\delta_1+\delta_2}{2})$. Therefore ξ is concave.

We now only need to show that ξ is continuous at 0. From the boundedness of $\ell(\mathbf{q},d)/v(\mathbf{q})$, we know that $\Delta H_{\ell,W,T,v}(z) = +\infty$ when $z > \sup \Delta\ell(\mathbf{q},d)/v(\mathbf{q})$. Therefore $\exists a > 0$ and $b \in R$ such that $a\varepsilon + b \leq \Delta H_{\ell,W,T,v}(\varepsilon)$. Now Pick any $\varepsilon' > 0$, and let $\delta' = \zeta_*(\varepsilon')/2$, we know from Proposition 25 that $\delta' > 0$. This implies that $\xi(\delta) < \varepsilon'$ when $\delta < \delta'$. ■

One can always choose $v(\mathbf{q}) \equiv 1$ to obtain a bound that applies to all underlying distributions on $\mathcal{X}$. However, with a more general v, one may obtain better bounds in some scenarios especially the low noise case. For example, see Theorem 13 in the main text.

Appendix B. Proof of Theorem 3

We shall first prove the following lemma.

Lemma 27 *$W^*(\mathbf{q}) := \inf_{\mathbf{f}\in\Omega} W(\mathbf{q},\mathbf{f})$ is a continuous function on Λ_K.*

Proof Consider a sequence $\mathbf{q}^{(m)} \in \Lambda_K$ such that $\lim_m \mathbf{q}^{(m)} = \mathbf{q}$. Without loss of generality, we assume that there exists k such that $\mathbf{q}_1 = \cdots = \mathbf{q}_k = 0$ and $\mathbf{q}_c > 0$ for $c > k$. Moreover, since each Ψ_c is bounded below, we may assume without loss of generality that $\Psi_c \geq 0$ (this condition can be achieved simply by adding a constant to each Ψ_c).

Now, let

$$\bar{W}(\mathbf{q}',\mathbf{f}) = \sum_{c=k+1}^{K} \mathbf{q}'_c \Psi_c(\mathbf{f})$$

and

$$\bar{W}^*(\mathbf{q}') = \inf_{\mathbf{f}\in\Omega} \sum_{c=k+1}^{K} \mathbf{q}'_c \Psi_c(\mathbf{f}).$$

Since $\{\bar{W}^*(\mathbf{q}^{(m)})\}_m$ is bounded, each sequence $\{\mathbf{q}_c^{(m)}\Psi_c(\cdot)\}_m$ is also bounded near the optimal solution. It is clear from the condition $\lim_m \mathbf{q}_c^{(m)} > 0$ $(c > k)$ that

$$\lim_{m\to\infty} \bar{W}^*(\mathbf{q}^{(m)}) = \bar{W}^*(\mathbf{q}) = W^*(\mathbf{q}).$$

Since $W^*(\mathbf{q}^{(m)}) \geq \bar{W}^*(\mathbf{q}^{(m)})$, we have

$$\liminf_{m\to\infty} W^*(\mathbf{q}^{(m)}) \geq W^*(\mathbf{q}). \tag{19}$$

Now for a sufficiently large positive number A, let

$$W_A^*(\mathbf{q}') = \inf_{\mathbf{f}\in\Omega, \|\mathbf{f}\|_1 \leq A} \sum_{c=1}^{K} \mathbf{q}'_c \Psi_c(\mathbf{f}).$$

We have

$$\limsup_{m\to\infty} W^*(\mathbf{q}^{(m)}) \leq \limsup_{m\to\infty} W_A^*(\mathbf{q}^{(m)}) = W_A^*(\mathbf{q}).$$

Since $\lim_{A\to\infty} W_A^*(\mathbf{q}) = W^*(\mathbf{q})$, we have

$$\limsup_{m\to\infty} W^*(\mathbf{q}^{(m)}) \le W^*(\mathbf{q}).$$

Combining this inequality with (19), we obtain the lemma. ■

Lemma 28 $\forall \varepsilon > 0$, $\exists \delta > 0$ *such that* $\forall \mathbf{q} \in \Lambda_k$:

$$\inf\left\{ W(\mathbf{q},\mathbf{f}) : \mathbf{f}_c = \sup_k \mathbf{f}_k, a_c\mathbf{q}_c \le \sup_k a_k\mathbf{q}_k - \varepsilon \right\} \ge W^*(\mathbf{q}) + \delta. \tag{20}$$

Proof We prove this by contradiction. Assume that (20) does not hold, then $\exists \varepsilon > 0$, and a sequence of $(c^{(m)}, \mathbf{f}^{(m)}, \mathbf{q}^{(m)})$ with $\mathbf{f}^{(m)} \in \Omega$ such that $\mathbf{f}^{(m)}_{c^{(m)}} = \sup_k \mathbf{f}^{(m)}_k$, $a_{c^{(m)}}\mathbf{q}^{(m)}_{c^{(m)}} \le \sup_k a_k\mathbf{q}^{(m)}_k - \varepsilon$, and

$$\lim_{m\to\infty} [W(\mathbf{q}^{(m)}, \mathbf{f}^{(m)}) - W^*(\mathbf{q}^{(m)})] = 0.$$

Since Λ_K is compact, we can choose a subsequence (which we still denoted as the whole sequence for simplicity) such that $c^{(m)} \equiv c^{(1)}$ and $\lim_m \mathbf{q}^{(m)} = \mathbf{q} \in \Lambda_K$. Using Lemma 27, we obtain

$$\lim_{m\to\infty} W(\mathbf{q}^{(m)}, \mathbf{f}^{(m)}) = W^*(\mathbf{q}).$$

Similar to the proof of Lemma 27, we assume that $\Psi_c \ge 0$ ($c = 1,\ldots,K$), $\mathbf{q}_1 = \cdots = \mathbf{q}_k = 0$ and $\mathbf{q}_c > 0$ ($c > k$). We obtain

$$\limsup_{m\to\infty} W(\mathbf{q}, \mathbf{f}^{(m)}) = \limsup_{m\to\infty} \sum_{c=k+1}^{K} \mathbf{q}_c^{(m)} \Psi_c(\mathbf{f}^{(m)}) \le \lim_{m\to\infty} W(\mathbf{q}^{(m)}, \mathbf{f}^{(m)}) = W^*(\mathbf{q}).$$

Note that our assumption also implies that $a_{c^{(1)}}\mathbf{q}_{c^{(1)}} \le \sup_k a_k\mathbf{q}_k - \varepsilon$ and $\mathbf{f}^{(m)}_{c^{(1)}} = \sup_k \mathbf{f}^{(m)}_k$. We have thus obtained a contradiction to the second ISC condition of $\Psi_c(\cdot)$. Therefore (20) must be valid. ■

Proof of the Theorem. We use the notations of Appendix A: let $\mathcal{X}$ be the input space, $\mathcal{Q} = \Lambda_K$ be the space of conditional probability vectors, and $\mathcal{D} = \{1,\ldots,K\}$ be the space of class labels. We let $\ell(\mathbf{q},k) = \sum_{c=1,c\neq k} a_c\mathbf{q}_c$, and thus the classification error of a decision function $p(\cdot)$ in (9) can be expressed as $\ell(p(\cdot)) = \mathbf{E}_X \ell([P(Y=c|X)]_c, p(X))$. The estimation-model space is $\Omega \subset R^K$, with decision T given by (5). The W function is given by (8). Let $v(\mathbf{q}) \equiv 1$. Then (20) implies that $\forall \varepsilon > 0, \Delta H_{\ell,W,T,v}(\varepsilon) > 0$. The theorem now follows directly from the claim of Corollary 26.

Appendix C. Infinite-Sample Inconsistency of the SVM Pairwise Comparison Method

Consider the non-differentiable SVM (hinge) loss $\phi(z) = (1-z)_+$. We show that the pairwise comparison method in (10) is not ISC with $K = 3$. More precisely, we have the following counter-example.

Proposition 29 *Let* $\mathbf{q} = [q_1, q_2, q_3]$ *with* $0 < q_3 < q_2 < q_1$ *such that* $q_1 < q_2 + q_3$ *and* $q_2 > 2q_3$. *Then* $W^*(\mathbf{q}) = W(\mathbf{q}, [1,1,0]) = 1 + q_1 + q_2 + 4q_3$.

Proof Consider $\mathbf{f} = [f_1, f_2, f_3]$. Without loss of generality, we can let $f_3 = 0$. Therefore

$$W(\mathbf{q}, \mathbf{f}) = 1 + q_1[\phi(f_1) + \phi(f_1 - f_2)] + q_2[\phi(f_2) + \phi(f_2 - f_1)] + q_3[\phi(-f_1) + \phi(-f_2)].$$

Clearly if $|f_1| > 100/q_3$ or $|f_2| > 100/q_3$, then $W(\mathbf{q}, \mathbf{f}) > 100 > W(\mathbf{q}, [0,0,0])$. Therefore the optimization of $W(\mathbf{q}, \mathbf{f})$ can be restricted to $|f_1|, |f_2| \leq 100/q_3$. It follows that $W^*(\mathbf{q})$ can be achieved at some point, still denote by $\mathbf{f} = [f_1, f_2, 0]$ such that $|f_1|, |f_2| \leq 100/q_3$.

From the order-preserving property of Theorem 5, we have $f_1 \geq f_2$, and $f_1, f_2 \geq f_3 = 0$. We can rewrite $W(\mathbf{q}, \mathbf{f})$ as

$$W(\mathbf{q}, \mathbf{f}) = 1 + q_1[\phi(f_1) + \phi(f_1 - f_2)] + q_2[\phi(f_2) + (f_1 - f_2) + 1] + q_3[f_1 + f_2 + 2].$$

If $f_2 < 1$, then

$$W(\mathbf{q}, [1 + f_1 - f_2, 1, 0]) - W(\mathbf{q}, [f_1, f_2, 0]) \leq -(q_2 - 2q_3)(1 - f_2) < 0.$$

Therefore we can assume that $f_1 \geq f_2 \geq 1$. Now

$$W(\mathbf{q}, \mathbf{f}) = 1 + q_1\phi(f_1 - f_2) + q_2[f_1 - f_2 + 1] + q_3[f_1 + f_2 + 2].$$

Since $q_1 < q_2 + q_3$, we have $q_1\phi(f_1 - f_2) + (q_2 + q_3)[f_1 - f_2] \geq q_1$, and the equality holds only when $f_1 = f_2$. Therefore $W(\mathbf{q}, \mathbf{f}) \geq 1 + q_1 + q_2[0 + 1] + q_3[2f_2 + 2]$, and the minimum can only be achieved at $f_1 = f_2 = 1$. ■

References

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. Technical Report 638, Statistics Department, University of California, Berkeley, 2003.

Gilles Blanchard, Olivier Bousquet, and Pascal Massart. Statistical performance of support vector machines. http://www.kyb.mpg.de/publications/pss/ps2731.ps, 2004.

Gilles Blanchard, Gabor Lugosi, and Nicolas Vayatis. On the rate of convergence of regularized boosting classifiers. *Journal of Machine Learning Research*, 4:861–894, 2003.

V. Blanz, V. Vapnik, and C. Burges. Multiclass discrimination with an extended support vector machine. Talk given at AT&T Bell Labs, 1995.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Ilya Desyatnikov and Ron Meir. Data-dependent bounds for multi-category classification based on convex losses. In *COLT*, 2003.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000. With discussion.

W. Jiang. Process consistency for adaboost. *The Annals of Statistics*, 32:13–29, 2004. with discussion.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of American Statistical Association*, 99:67–81, 2004.

Yi Lin. Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, pages 259–275, 2002.

Yufeng Liu and Xiaotong Shen. On multicategory ψ-learning and support vector machine. Private Communication, 2004.

G. Lugosi and N. Vayatis. On the Bayes-risk consistency of regularized boosting methods. *The Annals of Statistics*, 32:30–55, 2004. with discussion.

E. Mammen and A. Tsybakov. Smooth discrimination analysis. *Annals of Statis.*, 27:1808–1829, 1999.

Shie Mannor, Ron Meir, and Tong Zhang. Greedy algorithms for classification - consistency, convergence rates, and adaptivity. *Journal of Machine Learning Research*, 4:713–741, 2003.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, 2002.

Clint Scovel and Ingo Steinwart. Fast rates for support vector machines. Technical Report LA-UR 03-9117, Los Alamos National Laboratory, 2003.

Ingo Steinwart. Support vector machines are universally consistent. *J. Complexity*, 18:768–791, 2002.

Ingo Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4: 1071–1105, 2003.

Ingo Steinwart. Consistency of support vector machines and other regularized kernel machines. *IEEE Transactions on Information Theory*, 2004. to appear.

S. A. van de Geer. *Empirical Processes in M-estimation*. Cambridge University Press, 2000.

Aad W. van der Vaart and Jon 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.

Grace Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference series in applied mathematics. SIAM, 1990.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, 1998.

Tong Zhang. Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2:527–550, 2002.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statitics*, 32:56–85, 2004. with discussion.

Journal of Machine Learning Research 5 (2004) 1253–1286 Submitted 8/03; Revised 4/04; Published 10/04

The Minimum Error Minimax Probability Machine

Kaizhu Huang KZHUANG@CSE.CUHK.EDU.HK
Haiqin Yang HQYANG@CSE.CUHK.EDU.HK
Irwin King KING@CSE.CUHK.EDU.HK
Michael R. Lyu LYU@CSE.CUHK.EDU.HK
Laiwan Chan LWCHAN@CSE.CUHK.EDU.HK
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N. T., Hong Kong

Editor: Michael I. Jordan

Abstract

We construct a distribution-free Bayes optimal classifier called the Minimum Error Minimax Probability Machine (MEMPM) in a worst-case setting, i.e., under all possible choices of class-conditional densities with a given mean and covariance matrix. By assuming no specific distributions for the data, our model is thus distinguished from traditional Bayes optimal approaches, where an assumption on the data distribution is a must. This model is extended from the Minimax Probability Machine (MPM), a recently-proposed novel classifier, and is demonstrated to be the general case of MPM. Moreover, it includes another special case named the Biased Minimax Probability Machine, which is appropriate for handling biased classification. One appealing feature of MEMPM is that it contains an explicit performance indicator, i.e., a lower bound on the worst-case accuracy, which is shown to be tighter than that of MPM. We provide conditions under which the worst-case Bayes optimal classifier converges to the Bayes optimal classifier. We demonstrate how to apply a more general statistical framework to estimate model input parameters robustly. We also show how to extend our model to nonlinear classification by exploiting kernelization techniques. A series of experiments on both synthetic data sets and real world benchmark data sets validates our proposition and demonstrates the effectiveness of our model.

Keywords: classification, distribution-free, kernel, minimum error, sequential biased minimax probability machine, worst-case accuracies

1. Introduction

A novel model for two-category classification tasks called the Minimax Probability Machine (MPM) has been recently proposed (Lanckriet et al., 2002a). This model tries to minimize the probability of misclassification of future data points in a worst-case setting, i.e., under all possible choices of class-conditional densities with a given mean and covariance matrix. When compared with traditional generative models, MPM avoids making assumptions with respect to the data distribution; such assumptions are often invalid and lack generality. This model's performance is reported to be comparable to the Support Vector Machine (SVM) (Vapnik, 1999), a state-of-the-art classifier.

However, MPM forces the worst-case accuracies for two classes to be equal. This constraint seems inappropriate, since it is unnecessary that the worst-case accuracies are presumed equal.

Therefore, the classifier derived from this model does not result in minimizing the worst-case error rate of future data points and thus in a sense cannot represent the optimal classifier.

In this paper, by removing this constraint, we propose a generalized Minimax Probability Machine, called the Minimum Error Minimax Probability Machine (MEMPM). Instead of optimizing an equality-constrained worst-case error rate, this model minimizes the worst-case Bayes error rate of future data and thus achieves the optimum classifier in the worst-case scenario. Furthermore, this new model contains several appealing features.

First, as a generalized model, MEMPM includes and expands the Minimax Probability Machine. Interpretations from the viewpoints of the optimal thresholding problem and the geometry will be provided to show the advantages of MEMPM. Moreover, this generalized model includes another promising special case, named the Biased Minimax Probability Machine (BMPM) (Huang et al., 2004b), and extends its application to a type of important classification, i.e., biased classification.

Second, this model derives a distribution-free Bayes optimal classifier in the worst-case scenario. It thus distinguishes itself from the traditional Bayes optimal classifiers, which have to assume distributions for the data and thus lack generality in real cases. Furthermore, we will show that, under certain conditions, e.g., when a Gaussian distribution is assumed for the data, the worst-case Bayes optimal classifier becomes the true Bayes optimal hyperplane.

Third, similar to MPM, the MEMPM model also contains an explicit performance indicator, namely an explicit upper bound on the probability of misclassification of future data. Moreover, we will demonstrate theoretically and empirically that MEMPM attains a smaller upper bound of the probability of misclassification than MPM, which thus implies the superiority of MEMPM to MPM.

Fourth, although in general the optimization of MEMPM is shown to be a non-concave problem, empirically, it demonstrates reasonable concavity in the main "interest" region and thus can be solved practically. Furthermore, we will show that the final optimization problem involves solving a one-dimensional line search problem and thus results in a satisfactory solution.

This paper is organized as follows. In the next section, we present the main content of this paper, the MEMPM model, including its definition, interpretations, practical solving method, and sufficient conditions for convergence to the true Bayes decision hyperplane. Following that, we demonstrate a robust version of MEMPM. In Section 4, we seek to kernelize the MEMPM model to attack nonlinear classification problems. We then, in Section 5, present a series of experiments on synthetic data sets and real world benchmark data sets. In Section 6, we analyze the tightness of the worst-case accuracy bound. In Section 7, we show that empirically MEMPM is often concave in the main "interest" region. In Section 8, we present the limitations of MEMPM and envision possible future work. Finally, we conclude this paper in Section 9.

2. Minimum Error Minimax Probability Decision Hyperplane

In this section, we first present the model definition of MEMPM while reviewing the original MPM model. We then in Section 2.2 interpret MEMPM with respect to MPM. In Section 2.3, we specialize the MEMPM model for dealing with biased classification. In Section 2.4, we analyze the MEMPM optimization problem and propose a practical solving method. In Section 2.5, we address the sufficient conditions under which the worst-case Bayes optimal classifier derived from MEMPM becomes the true Bayes optimal classifier. In Section 2.6, we provide a geometrical interpretation for BMPM and MEMPM.

2.1 Problem Definition

The notation in this paper will largely follow that of Lanckriet et al. (2002b). Let $\mathbf{x}$ and $\mathbf{y}$ denote two random vectors representing two classes of data with means and covariance matrices as $\{\overline{\mathbf{x}}, \Sigma_{\mathbf{x}}\}$ and $\{\overline{\mathbf{y}}, \Sigma_{\mathbf{y}}\}$, respectively, in a two-category classification task, where $\mathbf{x}$, $\mathbf{y}$, $\overline{\mathbf{x}}$, $\overline{\mathbf{y}} \in \mathbb{R}^n$, and $\Sigma_{\mathbf{x}}$, $\Sigma_{\mathbf{y}} \in \mathbb{R}^{n\times n}$.

Assuming $\{\overline{\mathbf{x}}, \Sigma_{\mathbf{x}}\}$, $\{\overline{\mathbf{y}}, \Sigma_{\mathbf{y}}\}$ for two classes of data are reliable, MPM attempts to determine the hyperplane $\mathbf{a}^T\mathbf{z} = b$ ($\mathbf{a} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$, $\mathbf{z} \in \mathbb{R}^n$, $b \in \mathbb{R}$, and superscript T denotes the transpose) which can separate two classes of data with the maximal probability. The formulation for the MPM model is written as follows:

$$\begin{aligned} \max_{\alpha, \mathbf{a}\neq\mathbf{0}, b} \quad & \alpha \quad \text{s.t.} \\ & \inf_{\mathbf{x}\sim(\overline{\mathbf{x}}, \Sigma_{\mathbf{x}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{x} \geq b\} \geq \alpha, \\ & \inf_{\mathbf{y}\sim(\overline{\mathbf{y}}, \Sigma_{\mathbf{y}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{y} \leq b\} \geq \alpha, \end{aligned}$$

where α represents the lower bound of the accuracy for future data, namely, the worst-case accuracy. Future points $\mathbf{z}$ for which $\mathbf{a}^T\mathbf{z} \geq b$ are then classified as the class $\mathbf{x}$; otherwise they are judged as the class $\mathbf{y}$. This derived decision hyperplane is claimed to minimize the worst-case (maximal) probability of misclassification, or the error rate, of future data. Furthermore, this problem can be transformed to a convex optimization problem, or more specifically, a Second Order Cone Programming problem (Lobo et al., 1998; Nesterov and Nemirovsky, 1994).

As observed from the above formulation, this model assumes that the worst-case accuracies for two classes are the same. However, this assumption seems inappropriate, since it is unnecessary to require that the worst-case accuracies for two classes are exactly the same. Thus, the decision hyperplane given by this model does not necessarily minimize the worst-case error rate of future data and is not optimal in this sense. Motivated from the finding, we eliminate this constraint and propose a generalized model, the Minimum Error Minimax Probability Machine, as follows:

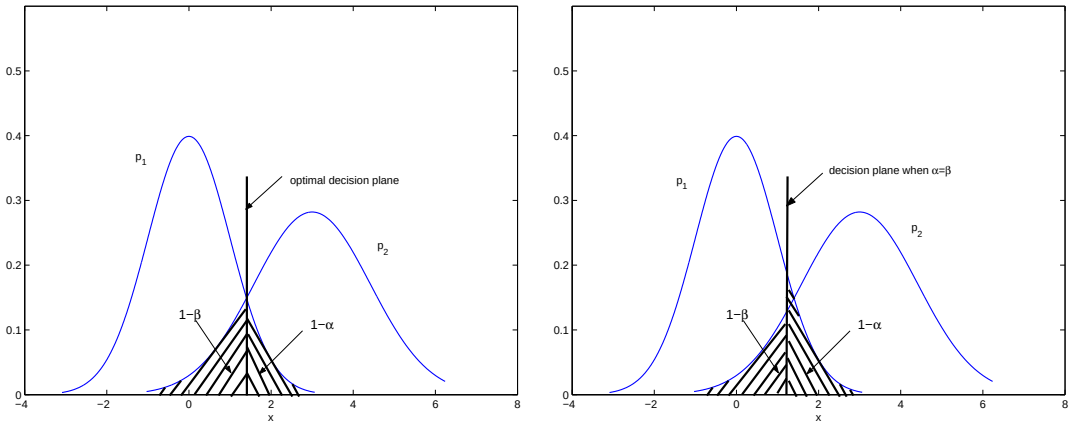
$$\begin{aligned} \max_{\alpha, \beta, \mathbf{a}\neq\mathbf{0}, b} \quad & \theta\alpha + (1-\theta)\beta \quad \text{s.t.} && (1) \\ & \inf_{\mathbf{x}\sim(\overline{\mathbf{x}}, \Sigma_{\mathbf{x}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{x} \geq b\} \geq \alpha, && (2) \\ & \inf_{\mathbf{y}\sim(\overline{\mathbf{y}}, \Sigma_{\mathbf{y}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{y} \leq b\} \geq \beta. && (3) \end{aligned}$$

Similarly, α and β indicate the worst-case classification accuracies of future data points for the class $\mathbf{x}$ and $\mathbf{y}$, respectively, while $\theta \in [0,1]$ is the prior probability of the class $\mathbf{x}$ and $1-\theta$ is thus the prior probability of the class $\mathbf{y}$. Intuitively, maximizing $\theta\alpha + (1-\theta)\beta$ can naturally be considered as maximizing the expected worst-case accuracy for future data. In other words, this optimization leads to minimizing the expected upper bound of the error rate. More precisely, if we change $\max\{\theta\alpha + (1-\theta)\beta\}$ to $\min\{\theta(1-\alpha) + (1-\theta)(1-\beta)\}$ and consider $1-\alpha$ as the upper bound probability that an $\mathbf{x}$ data point is classified as the class $\mathbf{y}$ ($1-\beta$ is similarly considered), the MEMPM model exactly minimizes the maximum Bayes error and thus derives the Bayes optimal hyperplane in the worst-case scenario.

2.2 Interpretation

We interpret MEMPM with respect to MPM in this section. First, it is evident that if we presume $\alpha = \beta$, the optimization of MEMPM degrades to the MPM optimization. Therefore, MPM is a special case of MEMPM.

An analogy to illustrate the difference between MEMPM and MPM can be seen in the optimal thresholding problem. Figure 1 illustrates this analogy. To separate two classes of one-dimensional data with density functions as p_1 and p_2, respectively, the optimal thresholding is given by the decision plane in Figure 1(a) (assuming the prior probabilities for two classes of data are equal). This optimal thesholding corresponds to the point minimizing the error rate $(1-\alpha)+(1-\beta)$ or maximizing the accuracy $\alpha+\beta$, which is exactly the intersection point of two density functions ($1-\alpha$ represents the area of 135^o-line filled region and $1-\beta$ represents the area of 45^o-line filled region). On the other hand, the thresholding point to force $\alpha = \beta$ is not necessarily the optimal point to separate these two classes.



(a) (b)

Figure 1: An analogy to illustrate the difference between MEMPM and MPM with equal prior probabilities for two classes. The optimal decision plane corresponds to the intersection point, where the error $(1-\alpha)+(1-\beta)$ is minimized (or the accuracy $\alpha+\beta$ is maximized) as implied by MEMPM, rather than the one, where α is equal to β as implied by MPM.

It should be clarified that the MEMPM model assumes no distributions. This distinguishes the MEMPM model from the traditional Bayes optimal methods, which have to make specific assumptions on the data distribution. On the other hand, although MEMPM minimizes the upper bound of the Bayes error rate of future data points, as shown later in Section 2.5, it will represent the true Bayes optimal hyperplane under certain conditions, in particular, when Gaussianity is assumed for the data.

2.3 Special Case for Biased Classification

The above discussion only covers unbiased classification, which does not favor one class over the other class intentionally. However, another important type of pattern recognition tasks, namely biased classification, arises very often in practice. In this scenario, one class is usually more important than the other class. Thus a bias should be imposed towards the important class. Such typical example can be seen in the diagnosis of epidemical disease. Classifying a patient who is infected with a disease into the opposite class results in serious consequences. Thus in this problem, the classification accuracy should be biased towards the class with disease. In other words, we would prefer to diagnose the person without the disease to be the infected case rather than the other way round.

In the following we demonstrate that MEMPM contains a special case we call the Biased Minimax Probability Machine for biased classification. We formulate this special case as

$$\begin{aligned} \max_{\alpha,\beta,\mathbf{a}\neq\mathbf{0},b} \quad & \alpha \quad \text{s.t.} \\ & \inf_{\mathbf{x}\sim(\bar{\mathbf{x}},\Sigma_{\mathbf{x}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{x} \geq b\} \geq \alpha, \\ & \inf_{\mathbf{y}\sim(\bar{\mathbf{y}},\Sigma_{\mathbf{y}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{y} \leq b\} \geq \beta_0, \end{aligned}$$

where $\beta_0 \in [0,\ 1)$, a pre-specified constant, represents an acceptable accuracy level for the less important class $\mathbf{y}$.

The above optimization utilizes a typical setting in biased classification, i.e., the accuracy for the important class (associated with $\mathbf{x}$) should be as high as possible, if only the accuracy for the less important class (associated with $\mathbf{y}$) maintains at an acceptable level specified by the lower bound β_0 (which can be set by users).

By quantitatively plugging a specified bias β_0 into classification and also by containing an explicit accuracy bound α for the important class, BMPM provides a direct and elegant means for biased classification. Comparatively, to achieve a specified bias, traditional biased classifiers such as the Weighted Support Vector Machine (Osuna et al., 1997) and the Weighted k-Nearest Neighbor method (Maloof et al., 2003) usually adapt different costs for different classes. However, due to the difficulties in establishing quantitative connections between the costs and the accuracy,[1] for imposing a specified bias, these methods have to resort to trial and error procedure to attain suitable costs; these procedures are generally indirect and lack rigorous treatments.

2.4 Solving the MEMPM Optimization Problem

In this section, we will propose to solve the MEMPM optimization problem. As will be demonstrated shortly, the MEMPM optimization can be transformed to a one-dimensional line search problem. More specifically, the objective function of the line search problem is implicitly determined by dealing with a BMPM problem. Therefore, solving the line search problem corresponds to solving a Sequential Biased Minimax Probability Machine (SBMPM) problem. Before we proceed, we first introduce how to solve the BMPM optimization problem.

1. Although cross validation might be used to provide empirical connections, they are problem-dependent and are usually slow procedures as well.

2.4.1 SOLVING THE BMPM OPTIMIZATION PROBLEM

First, we borrow Lemma 1 from Lanckriet et al. (2002b).

Lemma 1 *Given* $\mathbf{a} \neq \mathbf{0}$ *and b, such that* $\mathbf{a}^T\mathbf{y} \leq b$ *and* $\beta \in [0,1)$*, the condition*

$$\inf_{\mathbf{y}\sim(\overline{\mathbf{y}},\Sigma_\mathbf{y})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{y} \leq b\} \geq \beta,$$

holds if and only if $b - \mathbf{a}^T\overline{\mathbf{y}} \geq \kappa(\beta)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}$ *with* $\kappa(\beta) = \sqrt{\frac{\beta}{1-\beta}}$.

By using Lemma 1, we can transform the BMPM optimization problem as follows:

$$\begin{aligned} \max_{\alpha,\mathbf{a}\neq\mathbf{0},b} \quad & \alpha \quad \text{s.t.} \\ & -b + \mathbf{a}^T\overline{\mathbf{x}} \geq \kappa(\alpha)\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}\,, \qquad (4) \\ & b - \mathbf{a}^T\overline{\mathbf{y}} \geq \kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}\,, \qquad (5) \end{aligned}$$

where $\kappa(\alpha) = \sqrt{\frac{\alpha}{1-\alpha}}$, $\kappa(\beta_0) = \sqrt{\frac{\beta_0}{1-\beta_0}}$. (5) is directly obtained from (3) by using Lemma 1. Similarly, by changing $\mathbf{a}^T\mathbf{x} \geq b$ to $\mathbf{a}^T(-\mathbf{x}) \leq -b$, (4) can be obtained from (2).

From (4) and (5), we get

$$\mathbf{a}^T\overline{\mathbf{y}} + \kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}} \leq b \leq \mathbf{a}^T\overline{\mathbf{x}} - \kappa(\alpha)\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}\,.$$

If we eliminate *b* from this inequality, we obtain

$$\mathbf{a}^T(\overline{\mathbf{x}} - \overline{\mathbf{y}}) \geq \kappa(\alpha)\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}} + \kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}\,. \qquad (6)$$

We observe that the magnitude of $\mathbf{a}$ does not influence the solution of (6). Moreover, we can assume $\overline{\mathbf{x}} \neq \overline{\mathbf{y}}$; otherwise, the minimax machine does not have a physical meaning. In this case, (6) may even have no solution for every $\beta_0 \neq 0$, since the right hand side would always be positive provided that $\mathbf{a} \neq \mathbf{0}$. Thus in the extreme case, α and β have to be zero, implying that the worst-case classification accuracy is always zero.

Without loss of generality, we can set $\mathbf{a}^T(\overline{\mathbf{x}} - \overline{\mathbf{y}}) = 1$. Thus the problem can further be changed to:

$$\begin{aligned} \max_{\alpha,\mathbf{a}\neq\mathbf{0}} \quad & \alpha \quad \text{s.t.} \\ & 1 \geq \kappa(\alpha)\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}} + \kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}\,, \qquad (7) \\ & \mathbf{a}^T(\overline{\mathbf{x}} - \overline{\mathbf{y}}) = 1. \end{aligned}$$

Since $\Sigma_\mathbf{x}$ can be assumed to be positive definite (otherwise, we can always add a small positive amount to its diagonal elements and make it positive definite), from (7) we can obtain:

$$\kappa(\alpha) \leq \frac{1 - \kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}\,.$$

Because $\kappa(\alpha)$ increases monotonically with α, maximizing α is equivalent to maximizing $\kappa(\alpha)$, which further leads to

$$\max_{\mathbf{a}\neq\mathbf{0}} \quad \frac{1-\kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}} \quad \text{s.t.} \quad \mathbf{a}^T(\overline{\mathbf{x}}-\overline{\mathbf{y}})=1.$$

This kind of optimization is called the Fractional Programming (FP) problem (Schaible, 1995). To elaborate further, this optimization is equivalent to solving the following fractional problem:

$$\max_{\mathbf{a}\neq\mathbf{0}} \quad \frac{f(\mathbf{a})}{g(\mathbf{a})}\,, \tag{8}$$

subject to $\mathbf{a}\in A=\{\mathbf{a}|\mathbf{a}^T(\overline{\mathbf{x}}-\overline{\mathbf{y}})=1\}$, where $f(\mathbf{a})=1-\kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}$, $g(\mathbf{a})=\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}$.

Theorem 2 *The Fractional Programming problem (8) associated with the BMPM optimization is a pseudo-concave problem, whose every local optimum is the global optimum.*

Proof It is easy to see that the domain A is a convex set on $\mathbb{R}^n$, and that $f(\mathbf{a})$ and $g(\mathbf{a})$ are differentiable on A. Moreover, since $\Sigma_\mathbf{x}$ and $\Sigma_\mathbf{y}$ can be both considered as positive definite matrices, $f(\mathbf{a})$ is a concave function on A and $g(\mathbf{a})$ is a convex function on A. Then $\frac{f(\mathbf{a})}{g(\mathbf{a})}$ is a concave-convex FP problem. Hence it is a pseudoconcave problem (Schaible, 1995). Therefore, every local maximum is the global maximum (Schaible, 1995). ■

To handle this specific FP problem, many methods such as the parametric method (Schaible, 1995), the dual FP method (Schaible, 1977; Craven, 1988), and the concave FP method (Craven, 1978) can be used. A typical Conjugate Gradient method (Bertsekas, 1999) in solving this problem has a worst-case $O(n^3)$ time complexity. Adding the time cost to estimate $\overline{\mathbf{x}}$, $\overline{\mathbf{y}}$, $\Sigma_\mathbf{x}$, and $\Sigma_\mathbf{y}$, the total cost for this method is $O(n^3+Nn^2)$, where N is the number of data points. This complexity is in the same order as the linear Support Vector Machines (Schölkopf and Smola, 2002) and the linear MPM (Lanckriet et al., 2002b).

In this paper, the Rosen gradient projection method (Bertsekas, 1999) is used to find the solution of this pseudo-concave FP problem, which is proved to converge to a local maximum with a worst-case linear convergence rate. Moreover, the local maximum will exactly be the global maximum in this problem.

2.4.2 SEQUENTIAL BMPM OPTIMIZATION METHOD FOR MEMPM

We now turn to solving the MEMPM problem. Similar to Section 2.4.1, we can base Lemma 1 to transform the MEMPM optimization as follows:

$$\max_{\alpha,\beta,\mathbf{a}\neq\mathbf{0},b} \quad \theta\alpha+(1-\theta)\beta \quad \text{s.t.}$$

$$-b+\mathbf{a}^T\overline{\mathbf{x}}\geq\kappa(\alpha)\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}\,, \tag{9}$$

$$b-\mathbf{a}^T\overline{\mathbf{y}}\geq\kappa(\beta)\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}\,. \tag{10}$$

Using an analysis similar to that in Section 2.4.1, we can further transform the above optimization to:

$$\max_{\alpha,\beta,\mathbf{a}\neq\mathbf{0}} \quad \theta\alpha+(1-\theta)\beta \quad \text{s.t.} \tag{11}$$

$$1 \geq \kappa(\alpha)\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}+\kappa(\beta)\sqrt{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}}\,, \tag{12}$$

$$\mathbf{a}^T(\overline{\mathbf{x}}-\overline{\mathbf{y}})=1. \tag{13}$$

In the following we provide a lemma to show that the MEMPM solution is attained on the boundary of the set formed by the constraints of (12) and (13).

Lemma 3 *The maximum value of* $\theta\alpha+(1-\theta)\beta$ *under the constraints of (12) and (13) is achieved when the right hand side of (12) is strictly equal to* 1.

Proof Assume the maximum is achieved when $1>\kappa(\beta)\sqrt{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}}+\kappa(\alpha)\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}$. A new solution constructed by increasing α or $\kappa(\alpha)$ a small positive amount,[2] and maintaining β, $\mathbf{a}$ unchanged will satisfy the constraints and will be a better solution. ■

By applying Lemma 3, we can transform the optimization problem (11) under the constraints of (12) and (13) as follows:

$$\max_{\beta,\mathbf{a}\neq\mathbf{0}} \quad \frac{\theta\kappa^2(\alpha)}{\kappa^2(\alpha)+1}+(1-\theta)\beta \quad \text{s.t.} \tag{14}$$

$$\mathbf{a}^T(\overline{\mathbf{x}}-\overline{\mathbf{y}})=1, \tag{15}$$

where $\kappa(\alpha)=\frac{1-\kappa(\beta)\sqrt{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}}$.

In (14), if we fix β to a specific value within $[0,1)$, the optimization is equivalent to maximizing $\frac{\kappa^2(\alpha)}{\kappa^2(\alpha)+1}$ and further equivalent to maximizing $\kappa(\alpha)$, which is exactly the BMPM problem. We can then update β according to some rules and repeat the whole process until an optimal β is found. This is also the so-called line search problem (Bertsekas, 1999). More precisely, if we denote the value of optimization as a function $f(\beta)$, the above procedure corresponds to finding an optimal β to maximize $f(\beta)$. Instead of using an explicit function as in traditional line search problems, the value of the function here is implicitly given by a BMPM optimization procedure.

Many methods can be used to solve the line search problem. In this paper, we use the Quadratic Interpolation (QI) method (Bertsekas, 1999). As illustrated in Figure 2, QI finds the maximum point by updating a three-point pattern $(\beta_1,\ \beta_2,\ \beta_3)$ repeatedly. The new β denoted by β_{new} is given by the quadratic interpolation from the three-point pattern. Then a new three-point pattern is constructed by β_{new} and two of β_1,β_2,β_3. This method can be shown to converge superlinearly to a local optimum point (Bertsekas, 1999). Moreover, as shown in Section 7, although MEMPM generally cannot guarantee its concavity, empirically it is often concave. Thus the local optimum will often be the global optimum in practice.

2. Since $\kappa(\alpha)$ increases monotonically with α, increasing α a small positive amount corresponds to increasing $\kappa(\alpha)$ a small positive amount.

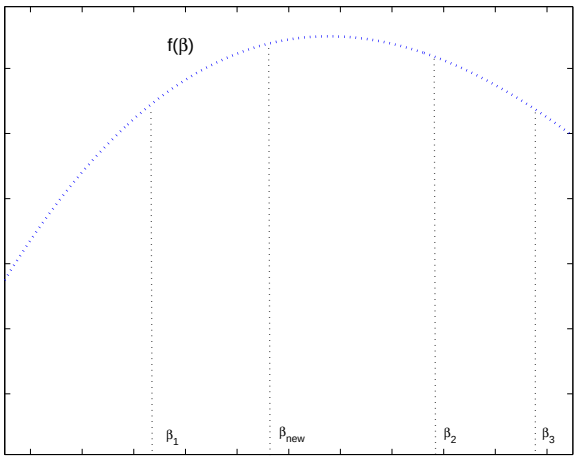


Figure 2: A three-point pattern and Quadratic Line search method. A β_{new} is obtained and a new three-point pattern is constructed by β_{new} and two of β_1, β_2 and β_3.

Until now, we do not mention how to calculate the intercept b. From Lemma 3, we can see that the inequalities (9) and (10) will become equalities at the maximum point $(\mathbf{a}_*, b_*)$. The optimal b will thus be obtained by

$$b_* = \mathbf{a}_*^T \overline{\mathbf{x}} - \kappa(\alpha_*)\sqrt{\mathbf{a}_*^T \Sigma_{\mathbf{x}} \mathbf{a}_*} = \mathbf{a}_*^T \overline{\mathbf{y}} + \kappa(\beta_*)\sqrt{\mathbf{a}_*^T \Sigma_{\mathbf{y}} \mathbf{a}_*}\,.$$

2.5 When Does the Worst-Case Bayes Optimal Hyperplane Become the True One?

As discussed, MEMPM derives the worst-case Bayes optimal hyperplane. Therefore, it is interesting to discover the conditions at which the worst-case optimal one changes to the true optimal one.

In the following we demonstrate two propositions. The first is that, when data are assumed to conform to some distributions, e.g., Gaussian distribution, the MEMPM framework leads to the Bayes optimal classifier; the second is that, when applied to high-dimensional classification tasks, the MEMPM model can be adapted to converge to the true Bayes optimal classifier under the Lyapunov condition.

To introduce the first proposition, we begin by assuming the data distribution as a Gaussian distribution.

Assuming $\mathbf{x} \sim \mathcal{N}(\overline{\mathbf{x}}, \Sigma_{\mathbf{x}})$ and $\mathbf{y} \sim \mathcal{N}(\overline{\mathbf{y}}, \Sigma_{\mathbf{y}})$, (2) becomes

$$\begin{aligned}
\inf_{\mathbf{x}\sim\mathcal{N}(\overline{\mathbf{x}},\Sigma_{\mathbf{x}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{x} \geq b\} &= \mathbf{Pr}_{\mathbf{x}\sim\mathcal{N}(\overline{\mathbf{x}},\Sigma_{\mathbf{x}})}\{\mathbf{a}^T\mathbf{x} \geq b\} \\
&= \mathbf{Pr}\{\mathcal{N}(0,1) \geq \frac{b - \mathbf{a}^T\overline{\mathbf{x}}}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}}\} \\
&= 1 - \Phi(\frac{b - \mathbf{a}^T\overline{\mathbf{x}}}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}}) \\
&= \Phi(\frac{-b + \mathbf{a}^T\overline{\mathbf{x}}}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}}) \geq \alpha, \qquad (16)
\end{aligned}$$

where $\Phi(z)$ is the cumulative distribution function for the standard normal Gaussian distribution:

$$\Phi(z) = \mathbf{Pr}\{\mathcal{N}(0,1) \leq z\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp(-s^2/2) ds.$$

Due to the monotonic property of $\Phi(z)$, we can further write (16) as

$$-b + \mathbf{a}^T \overline{\mathbf{x}} \geq \Phi^{-1}(\alpha) \sqrt{\mathbf{a}^T \Sigma_{\mathbf{x}} \mathbf{a}}\,.$$

Constraint (3) can be reformulated in a similar form. The optimization (1) is thus changed to:

$$\max_{\alpha,\beta,\mathbf{a}\neq\mathbf{0},b} \quad \theta\alpha + (1-\theta)\beta \quad \text{s.t.}$$

$$-b + \mathbf{a}^T \overline{\mathbf{x}} \geq \Phi^{-1}(\alpha) \sqrt{\mathbf{a}^T \Sigma_{\mathbf{x}} \mathbf{a}}\,, \tag{17}$$

$$b - \mathbf{a}^T \overline{\mathbf{y}} \geq \Phi^{-1}(\beta) \sqrt{\mathbf{a}^T \Sigma_{\mathbf{y}} \mathbf{a}}\,. \tag{18}$$

The above optimization is nearly the same as (1) subject to the constraints of (2) and (3) except that $\kappa(\alpha)$ is equal to $\Phi^{-1}(\alpha)$, instead of $\sqrt{\frac{\alpha}{1-\alpha}}$. Thus, it can similarly be solved based on the Sequential Biased Minimax Probability Machine method.

On the other hand, the Bayes optimal hyperplane corresponds to the one, $\mathbf{a}^T \mathbf{z} = b$ that minimizes the Bayes error:

$$\min_{\mathbf{a}\neq\mathbf{0},b} \quad \theta \mathbf{Pr}_{\mathbf{x}\sim\mathcal{N}(\overline{\mathbf{x}},\Sigma_{\mathbf{x}})}\{\mathbf{a}^T\mathbf{x} \leq b\} + (1-\theta)\mathbf{Pr}_{\mathbf{y}\sim\mathcal{N}(\overline{\mathbf{y}},\Sigma_{\mathbf{y}})}\{\mathbf{a}^T\mathbf{y} \geq b\}.$$

The above is exactly the upper bound of $\theta\alpha + (1-\theta)\beta$. From Lemma 3, we can know (17) and (18) will eventually become equalities. Traced back to (16), the equalities imply that α and β will achieve their upper bounds respectively. Therefore, when Gaussianity is assumed for the data, MEMPM derives the optimal Bayes hyperplane.

We propose Proposition 4 to extend the above analysis to general distribution assumptions.

Proposition 4 *If the distribution of the normalized random variable* $\frac{\mathbf{a}^T\mathbf{x} - \mathbf{a}^T\overline{\mathbf{x}}}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}}$*, denoted as* $\mathcal{NS}$*, is independent of* $\mathbf{a}$*, minimizing the Bayes error bound in MEMPM exactly minimizes the true Bayes error, provided that* $\Phi(z)$ *is changed to* $\mathbf{Pr}\{\mathcal{NS}(0,1) \leq z\}$.

Before presenting Proposition 6, we first introduce the central limit theorem under the Lyapunov condition (Chow and Teicher, 1997).

Theorem 5 *Let* $\mathbf{x}_n$ *be a sequence of independent random variables defined on the same probability space. Assume that* $\mathbf{x}_n$ *has finite expected value* μ_n *and finite standard deviation* σ_n*. We define* $s_n^2 = \sum_{i=1}^n \sigma_i^2$*. Assume that the Lyapunov conditions are satisfied, namely, the third central moment* $r_n^3 = \sum_{i=1}^n \mathrm{E}(|\mathbf{x}_n - \mu_n|^3)$ *is finite for every n, and that* $\lim_{n\to\infty} \frac{r_n}{s_n} = 0$*. The sum* $S_n = \mathbf{x}_1 + \ldots + \mathbf{x}_n$ *converges towards a Gaussian distribution.*

One interesting finding directly elicited from the above central limit theorem is that, if the component variable $\mathbf{x}_i$ of a given n-dimensional random variable $\mathbf{x}$ satisfies the Lyapunov condition, the sum of weighted component variables $\mathbf{x}_i$, $1 \leq i \leq n$, namely, $\mathbf{a}^T\mathbf{x}$ tends towards a Gaussian distribution, as n grows.[3] This shows that, under the Lyapunov condition, when the dimension n grows,

3. Some techniques such as Independent Component Analysis (Deco and Obradovic, 1996) can be applied to decorrelate the dependence among random variables beforehand.

the hyperplane derived by MEMPM with the Gaussianity assumption tends towards the true Bayes optimal hyperplane. In this case, the MEMPM using $\Phi^{-1}(\alpha)$, the inverse function of the normal cumulative distribution, instead of $\sqrt{\frac{\alpha}{1-\alpha}}$, will converge to the true Bayes optimal decision hyperplane in the high-dimensional space. We summarize the analysis in Proposition 6.

Proposition 6 *If the component variable $\mathbf{x}_i$ of a given n-dimensional random variable $\mathbf{x}$ satisfies the Lyapunov condition, the MEMPM hyperplane derived by using $\Phi^{-1}(\alpha)$, the inverse function of the normal cumulative distribution, will converge to the true Bayes optimal one.*

The underlying justifications in the above two propositions are rooted in the fact that the generalized MPM is exclusively determined by the first and second moments. These two propositions emphasize the dominance of the first and second moments in representing data. More specifically, Proposition 4 hints that the distribution is only decided by up to the second moments. The Lyapunov condition in Proposition 6 also implies that the second order moment dominates the third order moment in the long run. It is also noteworthy that, with a fixed mean and covariance, the distribution of Maximum Entropy Estimation is a Gaussian distribution (Keysers et al., 2002). This would once again suggest the usage of $\Phi^{-1}(\alpha)$ in the high-dimensional space.

2.6 Geometrical Interpretation

In this section, we first provide a parametric solving method for BMPM. We then demonstrate that this parametric method enables a nice geometrical interpretation for both BMPM and MEMPM.

2.6.1 A PARAMETRIC METHOD FOR BMPM

We present a parametric method to solve BMPM in the following. When compared with Gradient methods, this approach is relatively slow, but it need not calculate the gradient in each step and hence may avoid accumulated errors.

According to the parametric method, the fractional function can be iteratively optimized in two steps (Schaible, 1995):

Step 1: Find $\mathbf{a}$ by maximizing $f(\mathbf{a})-\lambda g(\mathbf{a})$ in the domain A, where $\lambda\in\mathbb{R}$ is the newly introduced parameter.

Step 2: Update λ by $\frac{f(\mathbf{a})}{g(\mathbf{a})}$.

The iteration of the above two steps will guarantee to converge to a local maximum, which is also the global maximum in our problem. In the following, we adopt a method to solve the maximization problem in Step 1. Replacing $f(\mathbf{a})$ and $g(\mathbf{a})$, we expand the optimization problem to:

$$\max_{\mathbf{a}\neq\mathbf{0}} \quad 1-\kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}}-\lambda\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}} \quad \text{s.t.} \quad \mathbf{a}^T(\overline{\mathbf{x}}-\overline{\mathbf{y}})=1. \tag{19}$$

Maximizing (19) is equivalent to $\min_{\mathbf{a}}\kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}}+\lambda\sqrt{\mathbf{a}^T\Sigma_{\mathbf{x}}\mathbf{a}}$ under the same constraint. By writing $\mathbf{a}=\mathbf{a}_0+\mathbf{F}\mathbf{u}$, where $\mathbf{a}_0=(\overline{\mathbf{x}}-\overline{\mathbf{y}})/\parallel\overline{\mathbf{x}}-\overline{\mathbf{y}}\parallel_2^2$ and $\mathbf{F}\in\mathbb{R}^{n\times(n-1)}$ is an orthogonal matrix whose columns span the subspace of vectors orthogonal to $\overline{\mathbf{x}}-\overline{\mathbf{y}}$, an equivalent form (a factor $\frac{1}{2}$ over each term has been dropped) to remove the constraint can be obtained:

$$\min_{\mathbf{u},\eta>0,\xi>0}\eta+\frac{\lambda^2}{\eta}\|{\Sigma_{\mathbf{x}}}^{1/2}(\mathbf{a}_0+\mathbf{F}\mathbf{u})\|_2^2+\xi+\frac{\kappa(\beta_0)^2}{\xi}\|{\Sigma_{\mathbf{y}}}^{1/2}(\mathbf{a}_0+\mathbf{F}\mathbf{u})\|_2^2\,,$$

where $\eta, \xi \in \mathbb{R}$. This optimization form is very similar to the one in the Minimax Probability Machine (Lanckriet et al., 2002a) and can also be solved by using an iterative least-squares approach.

2.6.2 A GEOMETRICAL INTERPRETATION FOR BMPM AND MEMPM

The parametric method enables a nice geometrical interpretation of BMPM and MEMPM in a fashion similar to that of MPM in Lanckriet et al. (2002b). Again, we assume $\overline{\mathbf{x}} \neq \overline{\mathbf{y}}$ for the meaningful classification and assume that $\Sigma_{\mathbf{x}}$ and $\Sigma_{\mathbf{y}}$ are positive definite for the purpose of simplicity.

By using the 2-norm definition of a vector $\mathbf{z} : \|\mathbf{z}\|_2 = \max\{\mathbf{u}^T\mathbf{z} : \|\mathbf{u}\|_2 \leq 1\}$, we can express (19) as its dual form:

$$\tau_* := \min_{\mathbf{a}\neq\mathbf{0}} \max_{\mathbf{u},\mathbf{v}} \quad \lambda\mathbf{u}^T\Sigma_{\mathbf{x}}^{1/2}\mathbf{a} + \kappa(\beta_0)\mathbf{v}^T\Sigma_{\mathbf{y}}^{1/2}\mathbf{a} + \tau(1 - \mathbf{a}^T(\overline{\mathbf{x}} - \overline{\mathbf{y}})) : \ \|\mathbf{u}\|_2 \leq 1, \|\mathbf{v}\|_2 \leq 1 \ .$$

We change the order of the min and max operators and consider the min:

$$\begin{aligned} &\min_{\mathbf{a}\neq\mathbf{0}} \quad \lambda\mathbf{u}^T\Sigma_{\mathbf{x}}^{1/2}\mathbf{a} + \kappa(\beta_0)\mathbf{v}^T\Sigma_{\mathbf{y}}^{1/2}\mathbf{a} + \tau(1 - \mathbf{a}^T(\overline{\mathbf{x}} - \overline{\mathbf{y}})) \\ &= \begin{cases} \tau & \text{if } \tau\overline{\mathbf{x}} - \lambda\Sigma_{\mathbf{x}}^{1/2}\mathbf{u} = \tau\overline{\mathbf{y}} + \kappa(\beta_0)\Sigma_{\mathbf{y}}^{1/2}\mathbf{v} \\ -\infty & \text{otherwise} \end{cases} . \end{aligned}$$

Thus, the dual problem can further be changed to:

$$\max_{\tau,\mathbf{u},\mathbf{v}} \quad \tau : \|\mathbf{u}\|_2 \leq 1, \|\mathbf{v}\|_2 \leq 1, \tau\overline{\mathbf{x}} - \lambda\Sigma_{\mathbf{x}}^{1/2}\mathbf{u} = \tau\overline{\mathbf{y}} + \kappa(\beta_0)\Sigma_{\mathbf{y}}^{1/2}\mathbf{v} \ .$$

By defining $\ell := 1/\tau$, we rewrite the dual problem as

$$\min_{\ell,\mathbf{u},\mathbf{v}} \quad \ell : \overline{\mathbf{x}} - \lambda\Sigma_{\mathbf{x}}^{1/2}\mathbf{u} = \overline{\mathbf{y}} + \kappa(\beta_0)\Sigma_{\mathbf{y}}^{1/2}\mathbf{v}, \|\mathbf{u}\|_2 \leq \ell, \|\mathbf{v}\|_2 \leq \ell \ . \tag{20}$$

When the optimum is attained, we have

$$\tau_* = \lambda\|\Sigma_{\mathbf{x}}^{1/2}\mathbf{a}_*\|_2 + \kappa(\beta_0)\|\Sigma_{\mathbf{y}}^{1/2}\mathbf{a}_*\|_2 = 1/\ell_* \ .$$

We consider each side of (20) as an ellipsoid centered at the mean $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ and shaped by the weighted covariance matrices $\lambda\Sigma_{\mathbf{x}}$ and $\kappa(\beta_0)\Sigma_{\mathbf{y}}$ respectively:

$$\mathcal{H}_{\mathbf{x}}(\ell) = \{\mathbf{x} = \overline{\mathbf{x}} + \lambda\Sigma_{\mathbf{x}}^{1/2}\mathbf{u} : \|\mathbf{u}\|_2 \leq \ell\}, \ \mathcal{H}_{\mathbf{y}}(\ell) = \{\mathbf{y} = \overline{\mathbf{y}} + \kappa(\beta_0)\Sigma_{\mathbf{y}}^{1/2}\mathbf{v} : \|\mathbf{v}\|_2 \leq \ell\} \ .$$

The above optimization involves finding a minimum ℓ for which two ellipsoids intersect. For the optimum ℓ, these two ellipsoids are tangential to each other. We further note that, according to Lemma 3, at the optimum, λ_*, which is maximized via a series of the above procedures, satisfies

$$\begin{aligned} &1 = \lambda_*\|\Sigma_{\mathbf{x}}^{1/2}\mathbf{a}_*\|_2 + \kappa(\beta_0)\|\Sigma_{\mathbf{y}}^{1/2}\mathbf{a}_*\|_2 = \tau_* = 1/\ell_* \\ &\Rightarrow \ell_* = 1 \ . \end{aligned}$$

This means that the ellipsoid for the class $\mathbf{y}$ finally changes to the one centered at $\overline{\mathbf{y}}$, whose Mahalanobis distance to $\overline{\mathbf{y}}$ is exactly equal to $\kappa(\beta_0)$. Moreover, the ellipsoid for the class $\mathbf{x}$ is the one centered at $\overline{\mathbf{x}}$ and tangential to the ellipsoid for the class $\mathbf{y}$. In comparison, for MPM, two ellipsoids

grow with the same speed (with the same $\kappa(\alpha)$ and $\kappa(\beta)$). On the other hand, since MEMPM corresponds to solving a sequence of BMPMs, it similarly leads to a hyperplane tangential to two ellipsoids, which achieves to minimize the maximum of the worst-case Bayes error. Moreover, it is not necessarily attained in a balanced way as in MPM, i.e., two ellipsoids do not necessarily grow with the same speed and hence probably contain the unequal Mahalanobis distance from their corresponding centers. This is illustrated in Figure 3.

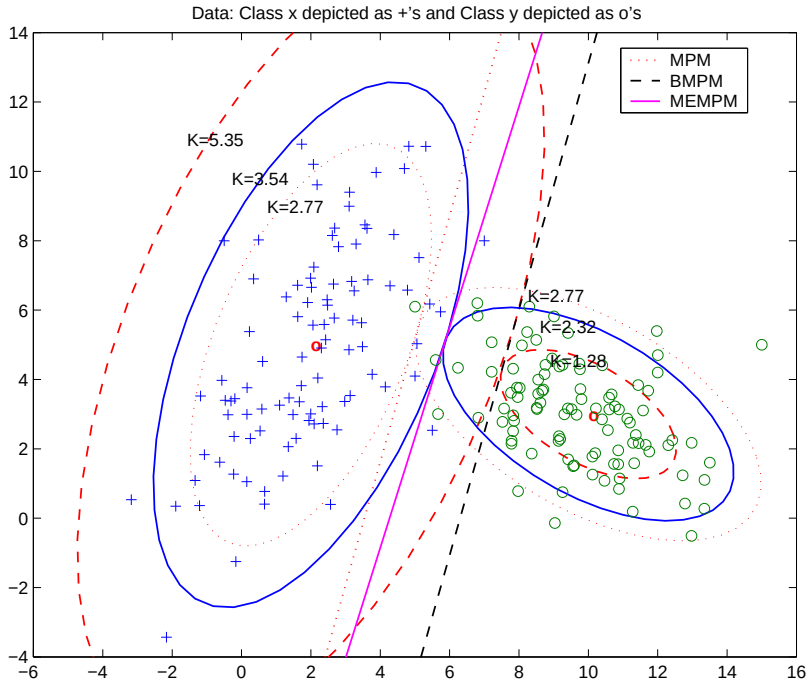


Figure 3: The geometrical interpretation of MEMPM and BMPM. Finding the optimal BMPM hyperplane corresponds to finding the decision plane (the black dashed line) tangential to an ellipsoid (the inner red dashed ellipsoid on the $\mathbf{y}$ side) , which is centered at $\overline{\mathbf{y}}$, shaped by the covariance $\Sigma_{\mathbf{y}}$ and whose Mahalanobis distance to $\overline{\mathbf{y}}$ is exactly equal to $\kappa(\beta_0)$ ($\kappa(\beta_0) = 1.28$ in this example). The worst-case accuracy α for $\mathbf{x}$ is determined by the Mahalanobis distance κ ($\kappa = 5.35$ in this example), at which, an ellipsoid (centered at $\overline{\mathbf{x}}$ and shaped by $\Sigma_{\mathbf{x}}$) is tangential to that $\kappa(\beta_0)$ ellipsoid, i.e., the outer red dashed ellipsoid on the $\mathbf{x}$ side. In comparison, MPM tries to find out the minimum equality-constrained κ, at which two ellipsoids for $\mathbf{x}$ and $\mathbf{y}$ intersect (both dotted red ellipsoids with $\kappa = 2.77$). For MEMPM, it achieves a tangent hyperplane in a non-balanced fashion, i.e., two ellipsoids may not attain the same κ but is globally optimal in the worst-case setting (see the solid blue ellipsoids).

3. Robust Version

In the above, the estimates of means and covariance matrices are assumed reliable. We now consider how the probabilistic framework in (1) changes in the face of variation of the means and covariance matrices:

$$\begin{aligned}
&\max_{\alpha,\beta,\mathbf{a}\neq\mathbf{0},b} \quad \theta\alpha+(1-\theta)\beta \quad \text{s.t.}\\
&\inf_{\mathbf{x}\sim(\bar{\mathbf{x}},\Sigma_{\mathbf{x}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{x}\geq b\}\geq\alpha, \forall(\bar{\mathbf{x}},\Sigma_{\mathbf{x}})\in\mathcal{X}\ ,\\
&\inf_{\mathbf{y}\sim(\bar{\mathbf{y}},\Sigma_{\mathbf{y}})} \mathbf{Pr}\{\mathbf{a}^T\mathbf{y}\leq b\}\geq\beta, \forall(\bar{\mathbf{y}},\Sigma_{\mathbf{y}})\in\mathcal{Y}\ ,
\end{aligned}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the sets of means and covariance matrices and are the subsets of $\mathbb{R}\times\mathcal{P}_n^+$, where $\mathcal{P}_n^+$ is the set of $n\times n$ symmetric positive semidefinite matrices.

Motivated by the tractability of the problem and from a statistical viewpoint, a specific setting of $\mathcal{X}$ and $\mathcal{Y}$ has been proposed in Lanckriet et al. (2002b). However, these authors consider the same variations of the means for two classes, which is easy to handle but less general. Now, considering the unequal treatment of each class, we propose the following setting, which is in a more general and complete form:

$$\begin{aligned}
\mathcal{X} &= \left\{(\bar{\mathbf{x}},\Sigma_{\mathbf{x}})\,|\,(\bar{\mathbf{x}}-\bar{\mathbf{x}}^0){\Sigma_{\mathbf{x}}}^{-1}(\bar{\mathbf{x}}-\bar{\mathbf{x}}^0)\leq\nu_{\mathbf{x}}^2,\ \|\Sigma_{\mathbf{x}}-{\Sigma_{\mathbf{x}}}^0\|_F\leq\rho_{\mathbf{x}}\right\},\\
\mathcal{Y} &= \left\{(\bar{\mathbf{y}},\Sigma_{\mathbf{y}})\,|\,(\bar{\mathbf{y}}-\bar{\mathbf{y}}^0){\Sigma_{\mathbf{y}}}^{-1}(\bar{\mathbf{y}}-\bar{\mathbf{y}}^0)\leq\nu_{\mathbf{y}}^2,\ \|\Sigma_{\mathbf{y}}-{\Sigma_{\mathbf{y}}}^0\|_F\leq\rho_{\mathbf{y}}\right\},
\end{aligned}$$

where $\bar{\mathbf{x}}^0$, $\Sigma_{\mathbf{x}}^0$ are the "nominal" mean and covariance matrices obtained through estimation. Parameters $\nu_{\mathbf{x}}$, $\nu_{\mathbf{y}}$, $\rho_{\mathbf{x}}$, and $\rho_{\mathbf{y}}$ are positive constants. The matrix norm is defined as the Frobenius norm: $\|M\|_F^2=\mathrm{Tr}(M^TM)$.

With the equality assumption for the variations of the means for two classes, the parameters $\nu_{\mathbf{x}}$ and $\nu_{\mathbf{y}}$ are required equal in Lanckriet et al. (2002b). This enables the direct usage of the MPM optimization in its robust version. However, the assumption may not be valid in real cases. Moreover, in MEMPM, the assumption is also unnecessary and inappropriate. This will be demonstrated later in the experiment.

By applying the results from Lanckriet et al. (2002b), we obtain the robust MEMPM as

$$\begin{aligned}
&\max_{\alpha,\beta,\mathbf{a}\neq\mathbf{0},b} \quad \theta\alpha+(1-\theta)\beta \quad \text{s.t.}\\
&-b+\mathbf{a}^T\bar{\mathbf{x}}^0\geq(\kappa(\alpha)+\nu_{\mathbf{x}})\sqrt{\mathbf{a}^T({\Sigma_{\mathbf{x}}}^0+\rho_{\mathbf{x}}I_n)\mathbf{a}},\\
&b-\mathbf{a}^T\bar{\mathbf{y}}^0\geq(\kappa(\beta)+\nu_{\mathbf{y}})\sqrt{\mathbf{a}^T({\Sigma_{\mathbf{y}}}^0+\rho_{\mathbf{y}}I_n)\mathbf{a}}.
\end{aligned}$$

Analogously, we transform the above optimization problem to

$$\max_{\alpha,\beta,\mathbf{a}\neq\mathbf{0}} \theta\frac{\kappa_r^2(\alpha)}{1+\kappa_r^2(\alpha)}+(1-\theta)\beta \quad \text{s.t. } \mathbf{a}^T(\bar{\mathbf{x}}^0-\bar{\mathbf{y}}^0)=1,$$

where $\kappa_r(\alpha)=\max\left(\frac{1-(\kappa(\beta)+\nu_{\mathbf{y}})\sqrt{\mathbf{a}^T({\Sigma_{\mathbf{y}}}^0+\rho_{\mathbf{y}}I_n)\mathbf{a}}}{\sqrt{\mathbf{a}^T({\Sigma_{\mathbf{x}}}^0+\rho_{\mathbf{x}}I_n)\mathbf{a}}}-\nu_{\mathbf{x}},0\right)$ and thus can be solved by the SBMPM method. The optimal b is therefore calculated by

$$\begin{aligned}
b_* &= {\mathbf{a}_*}^T\bar{\mathbf{x}}^0-(\kappa(\alpha_*)+\nu_{\mathbf{x}})\sqrt{{\mathbf{a}_*}^T({\Sigma_{\mathbf{x}}}^0+\rho_{\mathbf{x}}I_n)\mathbf{a}_*}\\
&= {\mathbf{a}_*}^T\bar{\mathbf{y}}^0+(\kappa(\beta_*)+\nu_{\mathbf{y}})\sqrt{{\mathbf{a}_*}^T({\Sigma_{\mathbf{y}}}^0+\rho_{\mathbf{y}}I_n)\mathbf{a}_*}.
\end{aligned}$$

Remarks. Interestingly, if MPM is treated with unequal robust parameters $\nu_{\mathbf{x}}$ and $\nu_{\mathbf{y}}$, it leads to solving an optimization similar to MEMPM, since $\kappa(\alpha)+\nu_{\mathbf{x}}$ will not be equal to $\kappa(\alpha)+\nu_{\mathbf{y}}$. In addition, similar to the robust MPM, when applied in practice, the specific values of $\nu_{\mathbf{x}}$, $\nu_{\mathbf{y}}$, $\rho_{\mathbf{x}}$, and $\rho_{\mathbf{y}}$ can be provided based on the central limit theorem or the resampling method.

4. Kernelization

We note that, in the above, the classifier derived from MEMPM is given in a linear configuration. In order to handle nonlinear classification problems, in this section, we seek to use the kernelization trick (Schölkopf and Smola, 2002) to map the *n*-dimensional data points into a high-dimensional feature space $\mathbb{R}^f$, where a linear classifier corresponds to a nonlinear hyperplane in the original space.

Since the optimization of MEMPM corresponds to a sequence of BMPM optimization problems, this model will naturally inherit the kernelization ability of BMPM. We thus in the following mainly address the kernelization of BMPM.

Assuming training data points are represented by $\{\mathbf{x}_i\}_{i=1}^{N_{\mathbf{x}}}$ and $\{\mathbf{y}_j\}_{j=1}^{N_{\mathbf{y}}}$ for the class $\mathbf{x}$ and class $\mathbf{y}$, respectively, the kernel mapping can be formulated as

$$\mathbf{x} \to \varphi(\mathbf{x}) \sim (\overline{\varphi(\mathbf{x})}, \Sigma_{\varphi(\mathbf{x})}),$$
$$\mathbf{y} \to \varphi(\mathbf{y}) \sim (\overline{\varphi(\mathbf{y})}, \Sigma_{\varphi(\mathbf{y})}),$$

where $\varphi : \mathbb{R}^n \to \mathbb{R}^f$ is a mapping function. The corresponding linear classifier in $\mathbb{R}^f$ is $\mathbf{a}^T\varphi(\mathbf{z}) = b$, where $\mathbf{a}$, $\varphi(\mathbf{z}) \in \mathbb{R}^f$, and $b \in \mathbb{R}$. Similarly, the transformed FP optimization in BMPM can be written as

$$\max_{\mathbf{a}} \frac{1-\kappa(\beta_0)\sqrt{\mathbf{a}^T\Sigma_{\varphi(\mathbf{y})}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_{\varphi(\mathbf{x})}\mathbf{a}}} \quad \text{s.t.} \quad \mathbf{a}^T(\overline{\varphi(\mathbf{x})} - \overline{\varphi(\mathbf{y})}) = 1. \tag{21}$$

However, to make the kernel work, we need to represent the final decision hyperplane and the optimization in a kernel form, $K(\mathbf{z}_1,\mathbf{z}_2) = \varphi(\mathbf{z}_1)^T\varphi(\mathbf{z}_2)$, namely an inner product form of the mapping data points.

4.1 Kernelization Theory for BMPM

In the following, we demonstrate that, although BMPM possesses a significantly different optimization form from MPM, the kernelization theory proposed in Lanckriet et al. (2002b) is still viable, provided that suitable estimates for means and covariance matrices are applied therein.

We first state a theory similar to Corollary 5 of Lanckriet et al. (2002b) and prove its validity in BMPM.

Corollary 7 *If the estimates of means and covariance matrices are given in BMPM as*

$$\overline{\varphi(\mathbf{x})} = \sum_{i=1}^{N_{\mathbf{x}}} \lambda_i \varphi(\mathbf{x}_i), \quad \overline{\varphi(\mathbf{y})} = \sum_{j=1}^{N_{\mathbf{y}}} \omega_j \varphi(\mathbf{y}_j),$$

$$\Sigma_{\varphi(\mathbf{x})} = \rho_{\mathbf{x}}\mathbf{I}_n + \sum_{i=1}^{N_{\mathbf{x}}} \Lambda_i(\varphi(\mathbf{x}_i) - \overline{\varphi(\mathbf{x})})(\varphi(\mathbf{x}_i) - \overline{\varphi(\mathbf{x})})^T ,$$

$$\Sigma_{\varphi(\mathbf{y})} = \rho_{\mathbf{y}}\mathbf{I}_n + \sum_{j=1}^{N_{\mathbf{y}}} \Omega_j(\varphi(\mathbf{y}_j) - \overline{\varphi(\mathbf{y})})(\varphi(\mathbf{y}_j) - \overline{\varphi(\mathbf{y})})^T ,$$

where $\mathbf{I}_n$ *is the identity matrix of dimension n, then the optimal* $\mathbf{a}$ *in problem (21) lies in the space spanned by the training points.*

Proof Similar to Lanckriet et al. (2002b), we write $\mathbf{a} = \mathbf{a}_p + \mathbf{a}_d$, where $\mathbf{a}_p$ is the projection of $\mathbf{a}$ in the vector space spanned by all the training data points and $\mathbf{a}_d$ is the orthogonal component to this span space. It can be easily verified that (21) changes to maximize the following:

$$\frac{1-\kappa(\beta_0)\sqrt{\mathbf{a}_p^T \sum_{i=1}^{N_{\mathbf{x}}} \Lambda_i(\varphi(\mathbf{x}_i) - \overline{\varphi(\mathbf{x})})(\varphi(\mathbf{x}_i) - \overline{\varphi(\mathbf{x})})^T \mathbf{w}_p + \rho_{\mathbf{x}}(\mathbf{a}_p^T\mathbf{a}_p + \mathbf{w}_d^T\mathbf{a}_d)}}{\sqrt{\mathbf{a}_p^T \sum_{j=1}^{N_{\mathbf{y}}} \Omega_j(\varphi(\mathbf{y}_j) - \overline{\varphi(\mathbf{y})})(\varphi(\mathbf{y}_j) - \overline{\varphi(\mathbf{y})})^T \mathbf{a}_p + \rho_{\mathbf{y}}(\mathbf{a}_p^T\mathbf{a}_p + \mathbf{a}_d^T\mathbf{a}_d)}}$$

subject to the constraints of $\mathbf{a}_p^T(\overline{\varphi(\mathbf{x})} - \overline{\varphi(\mathbf{y})}) = 1$.

Since we intend to maximize the fractional form and both the denominator and the numerator are positive, the denominator needs to be as small as possible and the numerator needs to be as large as possible. This would finally lead to $\mathbf{a}_d = \mathbf{0}$. In other words, the optimal $\mathbf{a}$ lies in the vector space spanned by all the training data points. Note that the introduction of $\rho_{\mathbf{x}}$ and $\rho_{\mathbf{y}}$ enables a direct application of the robust estimates in the kernelization. ■

According to Corollary 7, if appropriate estimates of means and covariance matrices are applied, the optimal $\mathbf{a}$ can be written as the linear combination of training points. In particular, if we obtain the means and covariance matrices as the plug-in estimates, i.e.,

$$\overline{\varphi(\mathbf{x})} = \frac{1}{N_{\mathbf{x}}} \sum_{i=1}^{N_{\mathbf{x}}} \varphi(\mathbf{x}_i) ,$$

$$\overline{\varphi(\mathbf{y})} = \frac{1}{N_{\mathbf{y}}} \sum_{j=1}^{N_{\mathbf{y}}} \varphi(\mathbf{y}_j) ,$$

$$\Sigma_{\varphi(\mathbf{x})} = \frac{1}{N_{\mathbf{x}}} \sum_{i=1}^{N_{\mathbf{x}}} (\varphi(\mathbf{x}_i) - \overline{\varphi(\mathbf{x})})(\varphi(\mathbf{x}_i) - \overline{\varphi(\mathbf{x})})^T ,$$

$$\Sigma_{\varphi(\mathbf{y})} = \frac{1}{N_{\mathbf{y}}} \sum_{j=1}^{N_{\mathbf{y}}} (\varphi(\mathbf{y}_j) - \overline{\varphi(\mathbf{y})})(\varphi(\mathbf{y}_j) - \overline{\varphi(\mathbf{y})})^T ,$$

we can write $\mathbf{a}$ as

$$\mathbf{a} = \sum_{i=1}^{N_{\mathbf{x}}} \mu_i \varphi(\mathbf{x}_i) + \sum_{j=1}^{N_{\mathbf{y}}} \upsilon_j \varphi(\mathbf{y}_j), \tag{22}$$

where the coefficients $\mu_i, \upsilon_j \in \mathbb{R}, i = 1,\ldots,N_{\mathbf{x}}, j = 1,\ldots,N_{\mathbf{y}}$.

By simply substituting (22) and four plug-in estimates into (21), we can obtain the Kernelization Theorem of BMPM.

Kernelization Theorem of BMPM *The optimal decision hyperplane of the problem (21) involves solving the Fractional Programming problem*

$$\kappa(\alpha_*) = \max_{\mathbf{w}\neq\mathbf{0}} \frac{1-\kappa(\beta_0)\sqrt{\frac{1}{N_\mathbf{y}}\mathbf{w}^T\tilde{\mathbf{K}}_\mathbf{y}^T\tilde{\mathbf{K}}_\mathbf{y}\mathbf{w}}}{\sqrt{\frac{1}{N_\mathbf{x}}\mathbf{w}^T\tilde{\mathbf{K}}_\mathbf{x}^T\tilde{\mathbf{K}}_\mathbf{x}\mathbf{w}}} \quad s.t. \quad \mathbf{w}^T(\tilde{\mathbf{k}}_\mathbf{x}-\tilde{\mathbf{k}}_\mathbf{y}) = 1\,. \tag{23}$$

The intercept b is calculated as

$$b_* = \mathbf{w}_*^T\tilde{\mathbf{k}}_\mathbf{x} - \kappa(\alpha_*)\sqrt{\frac{1}{N_\mathbf{x}}\mathbf{w}_*^T\tilde{\mathbf{K}}_\mathbf{x}^T\tilde{\mathbf{K}}_\mathbf{x}\mathbf{w}_*} = \mathbf{w}_*^T\tilde{\mathbf{k}}_\mathbf{y} + \kappa(\beta_0)\sqrt{\frac{1}{N_\mathbf{y}}\mathbf{w}_*^T\tilde{\mathbf{K}}_\mathbf{y}^T\tilde{\mathbf{K}}_\mathbf{y}\mathbf{w}_*}\,,$$

where $\kappa(\alpha_*)$ is obtained when (23) attains its optimum $(\mathbf{w}_*, b_*)$. For the robust version of BMPM, we can incorporate the variations of the means and covariances by conducting the following replacements:

$$\begin{aligned}
&\frac{1}{N_\mathbf{x}}\mathbf{w}_*^T\tilde{\mathbf{K}}_\mathbf{x}^T\tilde{\mathbf{K}}_\mathbf{x}\mathbf{w}_* \to \mathbf{w}_*^T(\frac{1}{N_\mathbf{x}}\tilde{\mathbf{K}}_\mathbf{x}^T\tilde{\mathbf{K}}_\mathbf{x} + \rho_\mathbf{x}\mathbf{K})\mathbf{w}_*\,,\\
&\frac{1}{N_\mathbf{y}}\mathbf{w}_*^T\tilde{\mathbf{K}}_\mathbf{y}^T\tilde{\mathbf{K}}_\mathbf{y}\mathbf{w}_* \to \mathbf{w}_*^T(\frac{1}{N_\mathbf{y}}\tilde{\mathbf{K}}_\mathbf{y}^T\tilde{\mathbf{K}}_\mathbf{y} + \rho_\mathbf{y}\mathbf{K})\mathbf{w}_*\,,\\
&\kappa(\beta_0) \to \kappa(\beta_0) + \mu_\mathbf{y}\,,\\
&\kappa(\alpha_*) \to \kappa(\alpha_*) + \mu_\mathbf{x}\,.
\end{aligned}$$

The optimal decision hyperplane can be represented as a linear form in the kernel space

$$f(\mathbf{z}) = \sum_{i=1}^{N_\mathbf{x}} \mathbf{w}_{*i}\mathbf{K}(\mathbf{z},\mathbf{x}_i) + \sum_{i=1}^{N_\mathbf{y}} \mathbf{w}_{*N_\mathbf{x}+i}\mathbf{K}(\mathbf{z},\mathbf{y}_i) - b_*.$$

The notation in the above are defined in Table 1.

5. Experiments

In this section, we first evaluate our model on a synthetic data set. Then we compare the performance of MEMPM with that of MPM, on six real world benchmark data sets. To demonstrate that BMPM is ideal for imposing a specified bias in classification, we also implement it on the Heart-disease data set. The means and covariance matrices for two classes are obtained directly from the training data sets by plug-in estimations. The prior probability θ is given by the proportion of $\mathbf{x}$ data in the training set.

5.1 Model Illustration on a Synthetic Data Set

To verify that the MEMPM model achieves the minimum Bayes error rate in the Gaussian distribution, we synthetically generate two classes of two-dimensional Gaussian data. As plotted in Figure 4(a), data associated with the class $\mathbf{x}$ are generated with the mean $\overline{\mathbf{x}}$ as $[3,0]^T$ and the covariance matrix $\Sigma_\mathbf{x}$ as $[4,\ 0; 0,\ 1]$, while data associated with the class $\mathbf{y}$ are generated with the mean $\overline{\mathbf{y}}$ as $[-1,0]^T$ and the covariance matrix $\Sigma_\mathbf{y}$ as $[1,\ 0; 0,\ 5]$. The solved decision hyperplane $Z_1 = 0.333$

Notation		
$\mathbf{z} \in \mathbb{R}^{N_\mathbf{x}+N_\mathbf{y}}$	$\mathbf{z}_i :=$	$\mathbf{x}_i \quad i = 1,2,\ldots,N_\mathbf{x}$.
	$\mathbf{z}_i :=$	$\mathbf{y}_{i-N_\mathbf{x}} \quad i = N_\mathbf{x}+1, N_\mathbf{x}+2, \ldots, N_\mathbf{x}+N_\mathbf{y}$.
$\mathbf{w} \in \mathbb{R}^{N_\mathbf{x}+N_\mathbf{y}}$	$\mathbf{w} :=$	$[\mu_1,\ldots,\mu_{N_\mathbf{x}},\upsilon_1,\ldots,\upsilon_{N_\mathbf{y}}]^T$.
$\mathbf{K}$ is Gram matrix	$\mathbf{K}_{i,j} :=$	$\varphi(\mathbf{z}_i)^T\varphi(\mathbf{z}_j)$.
	$\mathbf{K}_\mathbf{x} :=$	$\begin{pmatrix} \mathbf{K}_{1,1} & \mathbf{K}_{1,2} & \cdots & \mathbf{K}_{1,N_\mathbf{x}+N_\mathbf{y}} \\ \mathbf{K}_{2,1} & \mathbf{K}_{2,2} & \cdots & \mathbf{K}_{2,N_\mathbf{x}+N_\mathbf{y}} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{K}_{N_\mathbf{x},1} & \mathbf{K}_{N_\mathbf{x},2} & \cdots & \mathbf{K}_{N_\mathbf{x},N_\mathbf{x}+N_\mathbf{y}} \end{pmatrix}$.
	$\mathbf{K}_\mathbf{y} :=$	$\begin{pmatrix} \mathbf{K}_{N_\mathbf{x}+1,1} & \mathbf{K}_{N_\mathbf{x}+1,2} & \cdots & \mathbf{K}_{N_\mathbf{x}+1,N_\mathbf{x}+N_\mathbf{y}} \\ \mathbf{K}_{N_\mathbf{x}+2,1} & \mathbf{K}_{N_\mathbf{x}+2,2} & \cdots & \mathbf{K}_{N_\mathbf{x}+2,N_\mathbf{x}+N_\mathbf{y}} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{K}_{N_\mathbf{x}+N_\mathbf{y},1} & \mathbf{K}_{N_\mathbf{x}+N_\mathbf{y},2} & \cdots & \mathbf{K}_{N_\mathbf{x}+N_\mathbf{y},N_\mathbf{x}+N_\mathbf{y}} \end{pmatrix}$.
$\tilde{\mathbf{k}}_\mathbf{x}, \tilde{\mathbf{k}}_\mathbf{y} \in \mathbb{R}^{N_\mathbf{x}+N_\mathbf{y}}$	$[\tilde{\mathbf{k}}_\mathbf{x}]_i :=$	$\frac{1}{N_\mathbf{x}}\sum_{j=1}^{N_\mathbf{x}} \mathbf{K}(\mathbf{x}_j,\mathbf{z}_i)$.
	$[\tilde{\mathbf{k}}_\mathbf{y}]_i :=$	$\frac{1}{N_\mathbf{y}}\sum_{j=1}^{N_\mathbf{y}} \mathbf{K}(\mathbf{y}_j,\mathbf{z}_i)$.
$\mathbf{1}_{N_\mathbf{x}} \in \mathbb{R}^{N_\mathbf{x}}$	$\mathbf{1}_i :=$	$1 \quad i = 1,2,\ldots N_\mathbf{x}$.
$\mathbf{1}_{N_\mathbf{y}} \in \mathbb{R}^{N_\mathbf{y}}$	$\mathbf{1}_i :=$	$1 \quad i = 1,2,\ldots N_\mathbf{y}$.
$\tilde{\mathbf{K}} :=$	$\begin{pmatrix} \tilde{\mathbf{K}}_\mathbf{x} \\ \tilde{\mathbf{K}}_\mathbf{y} \end{pmatrix} :=$	$\begin{pmatrix} \mathbf{K}_\mathbf{x} - \mathbf{1}_{N_\mathbf{x}}\tilde{\mathbf{k}}_\mathbf{x}^T \\ \mathbf{K}_\mathbf{y} - \mathbf{1}_{N_\mathbf{y}}\tilde{\mathbf{k}}_\mathbf{y}^T \end{pmatrix}$.

Table 1: Notation used in Kernelization Theorem of BMPM

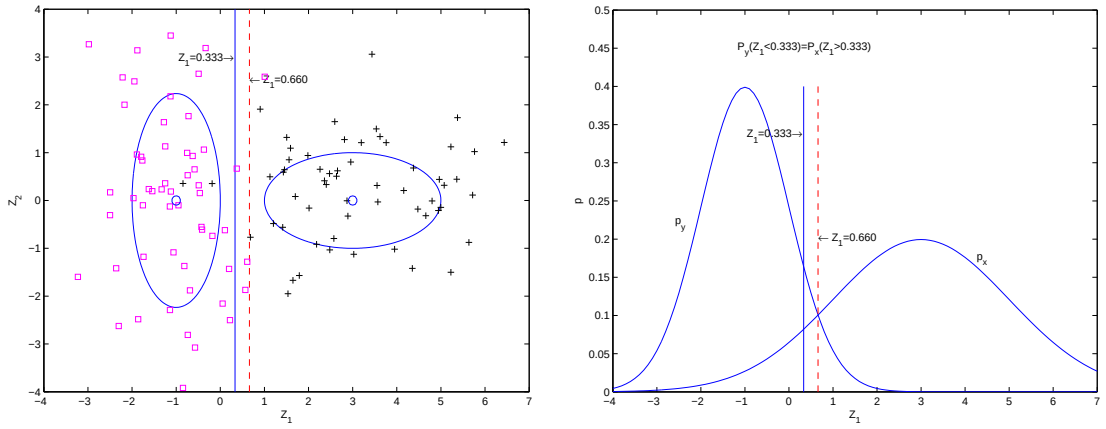
given by MPM is plotted as the solid blue line and the solved decision hyperplane $Z_1 = 0.660$ given by MEMPM is plotted as the dashed red line. From the geometrical interpretation, both hyperplanes should be perpendicular to the Z_1 axis.

As shown in Figure 4(b), the MEMPM hyperplane exactly represents the optimal thresholding under the distributions of the first dimension for two classes of data, i.e., the intersection point of two density functions. On the other hand, we find that, the MPM hyperplane exactly corresponds to the thresholding point with the same error rate for two classes of data, since the cumulative distributions $P_\mathbf{x}(Z_1 < 0.333)$ and $P_\mathbf{y}(Z_1 > 0.333)$ are exactly the same.

5.2 Evaluations on Benchmark Data Sets

We next evaluate our algorithm on six benchmark data sets. Data for Twonorm problem were generated according to Breiman (1997). The remaining five data sets (Breast, Ionosphere, Pima, Heart-disease, and Vote) were obtained from the UCI machine learning repository (Blake and Merz, 1998). Since handling the missing attribute values is out of the scope of this paper, we simply remove instances with missing attribute values in these data sets.

We randomly partition data into 90% training and 10% test sets. The final results are averaged over 50 random partitions of data. We compare the performance of MEMPM and MPM in both the linear setting and Gaussian kernel setting. The width parameter (σ) for the Gaussian kernel is obtained via cross validations over 50 random partitions of the training set. The experimental results are summarized in Table 2 and Table 3 for the linear kernel and Gaussian kernel respectively.



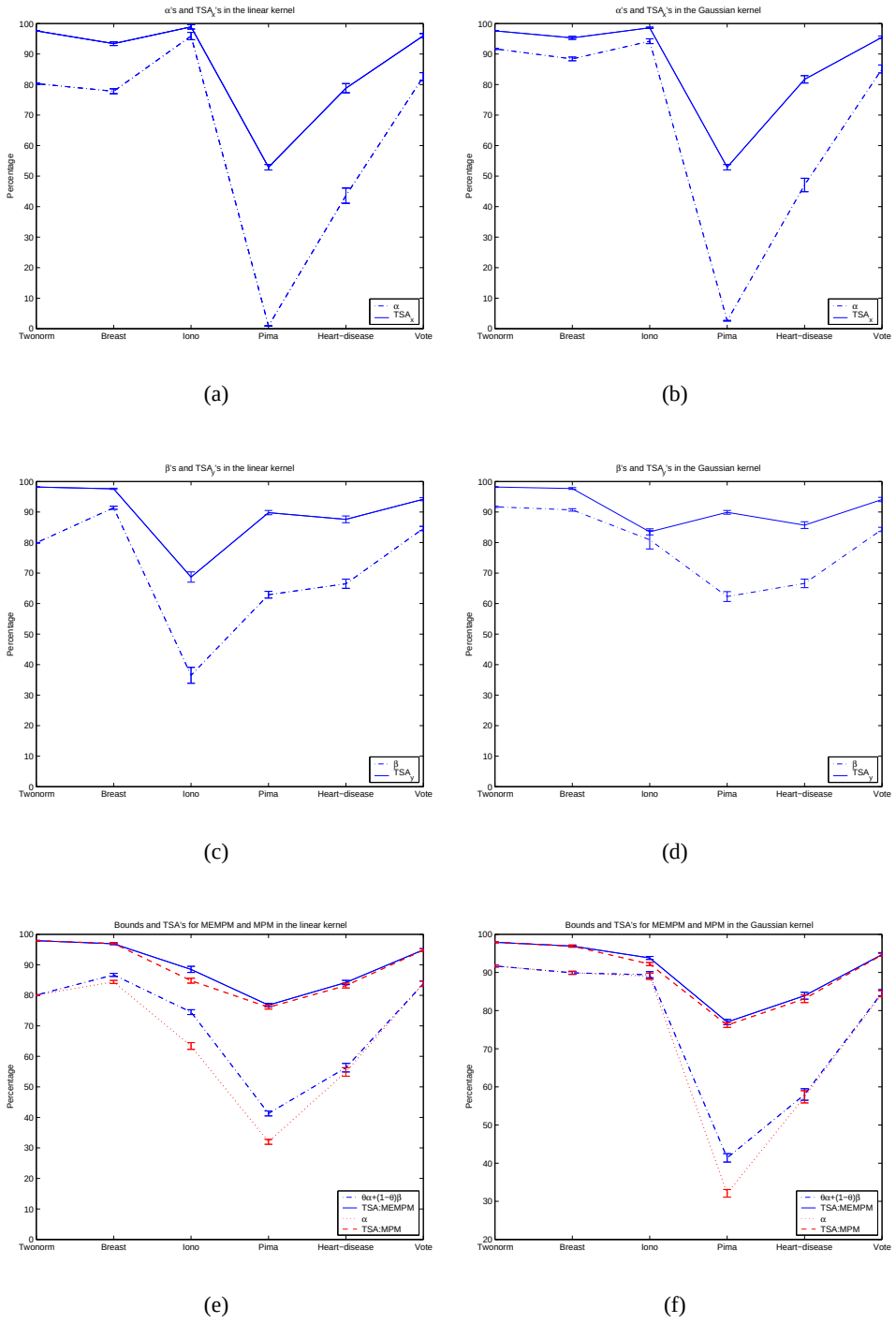
(a) (b)

Figure 4: An experiment on a synthetic data set. The decision hyperplane derived from MEMPM (the dashed red line) exactly corresponds to the optimal thresholding point, i.e., the intersection point, while the decision hyperplane given by MPM (the solid blue line) corresponds to the point in which the error rates for the two classes of data are equal.

From the results, we can see that our MEMPM demonstrates better performance than MPM in both the linear setting and Gaussian kernel setting. Moreover, in these benchmark data sets, the MEMPM hyperplanes are obtained with significantly unequal α and β except in Twonorm. This further confirms the validity of our proposition, i.e., the optimal minimax machine is not certain to achieve the same worst-case accuracies for two classes. Twonorm is not an exception to this. The two classes of data in Twonorm are generated under the multivariate normal distributions with the same covariance matrices. In this special case, the intersection point of two density functions will exactly represent the optimal thresholding point and the one with the same error rate for each class as well. Another important finding is that the accuracy bounds, namely $\theta\alpha+(1-\theta)\beta$ in MEMPM and α in MPM, are all increased in the Gaussian kernel setting when compared with those in the linear setting. This shows the advantage of the kernelized probability machine over the linear probability machine.

In addition, to show the relationship between the bounds and the test set accuracies (TSA) clearly, we plot them in Figure 5. As observed, the test set accuracies including $\text{TSA}_{\mathbf{x}}$ (for class $\mathbf{x}$), $\text{TSA}_{\mathbf{y}}$ (for the class $\mathbf{y}$), and the overall accuracies TSA are all greater than their corresponding accuracy bounds in both MPM and MEMPM. This demonstrates how the accuracy bound can serve as the performance indicator on future data. It is also observed that the overall worst-case accuracies $\theta\alpha+(1-\theta)\beta$ in MEMPM are greater than α in MPM both in the linear and Gaussian setting. This again demonstrates the superiority of MEMPM to MPM.

Since the lower bounds keep well within the test accuracies in the above experimental results, we do not perform the robust version of both models for the real world data sets. To see how the robust version works, we generate two classes of Gaussian data. As illustrated in Figure 6, $\mathbf{x}$ data are sampled from the Gaussian distribution with the mean as $[3,0]^T$ and the covariance as



(a) (b)

(c) (d)

(e) (f)

Figure 5: Bounds and test set accuracies. The test accuracies including $\text{TSA}_{\mathbf{x}}$ (for the class **x**), $\text{TSA}_{\mathbf{y}}$ (for the class **y**), and the overall accuracies TSA are all greater than their corresponding accuracy bounds in both MPM and MEMPM. This demonstrates how the accuracy bound can serve as the performance indicator on future data.

Data Set	MEMPM				MPM	
	α	β	$\theta\alpha+(1-\theta)\beta$	Accuracy	α	Accuracy
Twonorm(%)	$80.3\pm0.2\%$	$79.9\pm0.1\%$	$80.1\pm0.1\%$	$\mathbf{97.9\pm0.1}\%$	$80.1\pm0.1\%$	$\mathbf{97.9\pm0.1}\%$
Breast(%)	$77.8\pm0.8\%$	$91.4\pm0.5\%$	$86.7\pm0.5\%$	$96.9\pm0.3\%$	$84.4\pm0.5\%$	$\mathbf{97.0\pm0.2}\%$
Ionosphere(%)	$95.9\pm1.2\%$	$36.5\pm2.6\%$	$74.5\pm0.8\%$	$\mathbf{88.5\pm1.0}\%$	$63.4\pm1.1\%$	$84.8\pm0.8\%$
Pima(%)	$0.9\pm0.0\%$	$62.9\pm1.1\%$	$41.3\pm0.8\%$	$\mathbf{76.8\pm0.6}\%$	$32.0\pm0.8\%$	$76.1\pm0.6\%$
Heart-disease(%)	$43.6\pm2.5\%$	$66.5\pm1.5\%$	$56.3\pm1.4\%$	$\mathbf{84.2\pm0.7}\%$	$54.9\pm1.4\%$	$83.2\pm0.8\%$
Vote(%)	$82.6\pm1.3\%$	$84.6\pm0.7\%$	$83.9\pm0.9\%$	$\mathbf{94.9\pm0.4}\%$	$83.8\pm0.9\%$	$94.8\pm0.4\%$

Table 2: Lower bound α, β, and test accuracy compared to MPM in the linear setting.

Data Set	MEMPM				MPM	
	α	β	$\theta\alpha+(1-\theta)\beta$	Accuracy	α	Accuracy
Twonorm(%)	$91.7\pm0.2\%$	$91.7\pm0.2\%$	$91.7\pm0.2\%$	$\mathbf{97.9\pm0.1}\%$	$91.7\pm0.2\%$	$\mathbf{97.9\pm0.1}\%$
Breast(%)	$88.4\pm0.6\%$	$90.7\pm0.4\%$	$89.9\pm0.4\%$	$\mathbf{96.9\pm0.2}\%$	$89.9\pm0.4\%$	$\mathbf{96.9\pm0.3}\%$
Ionosphere(%)	$94.2\pm0.8\%$	$80.9\pm3.0\%$	$89.4\pm0.8\%$	$\mathbf{93.8\pm0.4}\%$	$89.0\pm0.8\%$	$92.2\pm0.4\%$
Pima(%)	$2.6\pm0.1\%$	$62.3\pm1.6\%$	$41.4\pm1.1\%$	$\mathbf{77.0\pm0.7}\%$	$32.1\pm1.0\%$	$76.2\pm0.6\%$
Heart-disease(%)	$47.1\pm2.2\%$	$66.6\pm1.4\%$	$58.0\pm1.5\%$	$\mathbf{83.9\pm0.9}\%$	$57.4\pm1.6\%$	$83.1\pm1.0\%$
Vote(%)	$85.1\pm1.3\%$	$84.3\pm0.7\%$	$84.7\pm0.8\%$	$\mathbf{94.7\pm0.5}\%$	$84.4\pm0.8\%$	$94.6\pm0.4\%$

Table 3: Lower bound α, β, and test accuracy compared to MPM with the Gaussian kernel.

$[1\ 0;0\ 3]$, while $\mathbf{y}$ data are sampled from another Gaussian distribution with the mean as $[-3,0]^T$ and the covariance as $[3\ 0;0\ 1]$. We randomly select 10 points of each class for training and leave the remaining points for test from the above synthetic data set. We present the result below.

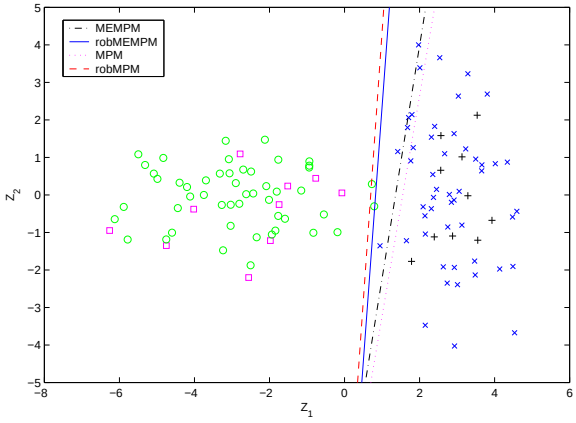
First, we calculate the corresponding means, $\bar{\mathbf{x}}^0$ and $\bar{\mathbf{y}}^0$ and covariance matrices, ${\Sigma_{\mathbf{x}}}^0$ and ${\Sigma_{\mathbf{y}}}^0$ and plug them into the linear MPM and the linear MEMPM. We obtain the MPM decision line (magenta dotted line) with a lower bound (assuming the Gaussian distribution) being 99.1% and the MEMPM decision line (black dash-dot line) with a lower bound of 99.7%. However, for the test set, we obtain the accuracies of only 93.0% for MPM and 97.0% for MEMPM (see Figure 6(a)). This obviously violates the lower bound.

Based on our knowledge of the real means and covariance matrices in this example, we set the parameters as

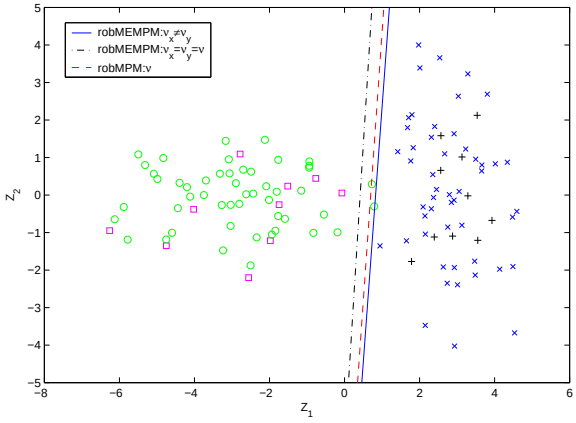
$$\nu_{\mathbf{x}} = \sqrt{(\bar{\mathbf{x}}-\bar{\mathbf{x}}^0)^T{\Sigma_{\mathbf{x}}}^{-1}(\bar{\mathbf{x}}-\bar{\mathbf{x}}^0)} = 0.046,\ \nu_{\mathbf{y}} = \sqrt{(\bar{\mathbf{y}}-\bar{\mathbf{y}}^0)^T{\Sigma_{\mathbf{y}}}^{-1}(\bar{\mathbf{y}}-\bar{\mathbf{y}}^0)} = 0.496,$$
$$\nu = \max(\nu_{\mathbf{x}},\nu_{\mathbf{y}}), \quad \rho_{\mathbf{x}} = \|\Sigma_{\mathbf{x}}-{\Sigma_{\mathbf{x}}}^0\|_F = 1.561, \quad \rho_{\mathbf{y}} = \|\Sigma_{\mathbf{y}}-{\Sigma_{\mathbf{y}}}^0\|_F = 0.972\ .$$

We then train the robust linear MPM and the robust linear MEMPM by these parameters and obtain the robust MPM decision line (red dashed line), and the robust MEMPM decision line (blue solid line), as seen in Figure 6(a). The lower bounds decrease to 87.3% for MPM and 93.2% for MEMPM respectively, but the test accuracies increase to 98.0% for MPM and 100.0% for MEMPM. Obviously, the lower bounds accord with the test accuracies.

Note that in the above, the robust MEMPM also achieves better performance than the robust MPM. Moreover, $\nu_{\mathbf{x}}$ and $\nu_{\mathbf{y}}$ are not necessarily the same. To see the result of MEMPM when $\nu_{\mathbf{x}}$ and $\nu_{\mathbf{y}}$ are forced to be the same, we train the robust MEMPM again by setting the parameters as $\nu_{\mathbf{x}} = \nu_{\mathbf{y}} = \nu$ as used in MPM. We obtain the corresponding decision line (black dash-dot line) as seen in Figure 6(b). The lower bound decreases to 91.0% and the test accuracy decreases to 98.0%. The above example indicates how the robust MEMPM clearly improves on the standard MEMPM when a bias is incorporated by inaccurate plug-in estimates and also validates that $\nu_{\mathbf{x}}$ need not be equal to $\nu_{\mathbf{y}}$.



(a)



(b)

Figure 6: An example in $\mathbb{R}^2$ demonstrates the results of robust versions of MEMPM and MPM. Training points are indicated with black $+$'s for the class $\mathbf{x}$ and magenta $\square$'s for the class $\mathbf{y}$. Test points are represented by blue $\times$'s for the class $\mathbf{x}$ and by green o's for the class $\mathbf{y}$. In (a), the robust MEMPM outperforms both MEMPM and the robust MPM. In (b), the robust MEMPM with $\nu_{\mathbf{x}} \neq \nu_{\mathbf{y}}$ outperforms the robust MEMPM with $\nu_{\mathbf{x}} = \nu_{\mathbf{y}}$.

5.3 Evaluations of BMPM on the Heart-Disease Data Set

To demonstrate the advantages of the BMPM model in dealing with biased classification, we implement BMPM on the Heart-disease data set, where a different treatment for different classes is necessary. The **x** class is associated with subjects with heart disease, whereas the **y** class corresponds to subjects without heart disease. Obviously, a bias should be considered for **x**, since misclassification of an **x** case into the opposite class would delay the therapy and may have a higher risk than the other way round. Similarly, we randomly partition data into 90% training and 10% test sets. Also, the width parameter (σ) for the Gaussian kernel is obtained via cross validations over 50 random partitions of the training set. We repeat the above procedures 50 times and report the average results.

By intentionally varying β_0 from 0 to 1, we obtain a series of test accuracies, including the **x** accuracy, $\text{TSA}_{\mathbf{x}}$, the **y** accuracy $\text{TSA}_{\mathbf{y}}$ for both the linear and Gaussian kernel. For simplicity, we denote the **x** accuracy in the linear setting as $\text{TSA}_{\mathbf{x}}(\text{L})$, while others are similarly defined.

The results are summarized in Figure 7. Four observations are worth highlighting. First, in both linear and Gaussian kernel settings, the smaller β_0 is, the higher the test accuracy for **x** becomes. This indicates that a bias can indeed be embedded in the classification boundary for the important class **x** by specifying a relatively smaller β_0. In comparison, MPM forces an equal treatment on each class and thus is not suitable for biased classification. Second, the test accuracies for **y** and **x** are strictly lower bounded by β_0 and α. This shows how a bias can be quantitatively, directly, and rigorously imposed towards the important class **x**. Note that again, for other weight-adapting based biased classifiers, the weights themselves lack accurate interpretations and thus cannot rigorously impose a specified bias, i.e., they would try different weights for a specified bias. Third, when given a prescribed β_0, the test accuracy for **x** and its worst-case accuracy α in the Gaussian kernel setting are both greater than the corresponding accuracies in the linear setting. Once again, this demonstrates the power of the kernelization. Fourth, we note that β_0 actually contains an upper bound, which is around 90% for the linear BMPM in this data set. This is reasonable. Observed from (7), the maximum β_0, denoted as β_{0_m}, is decided by setting $\alpha = 0$, i.e.,

$$\kappa(\beta_{0_m}) = \max_{\mathbf{a}\neq\mathbf{0}} \quad \frac{1}{\sqrt{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}}} \quad \text{s.t.} \quad \mathbf{a}^T(\overline{\mathbf{x}}-\overline{\mathbf{y}}) = 1.$$

It is interesting to note that when β_0 is set to zero, the test accuracies for **y** in the linear and Gaussian settings are both around 50% (see Figure 7(b)). This seeming "irrationality" is actually reasonable. We will discuss this in the next section.

6. How Tight Is the Bound?

A natural question for MEMPM is, how tight is the worst-case bound? In this section, we present a theoretical analysis in addressing this problem.

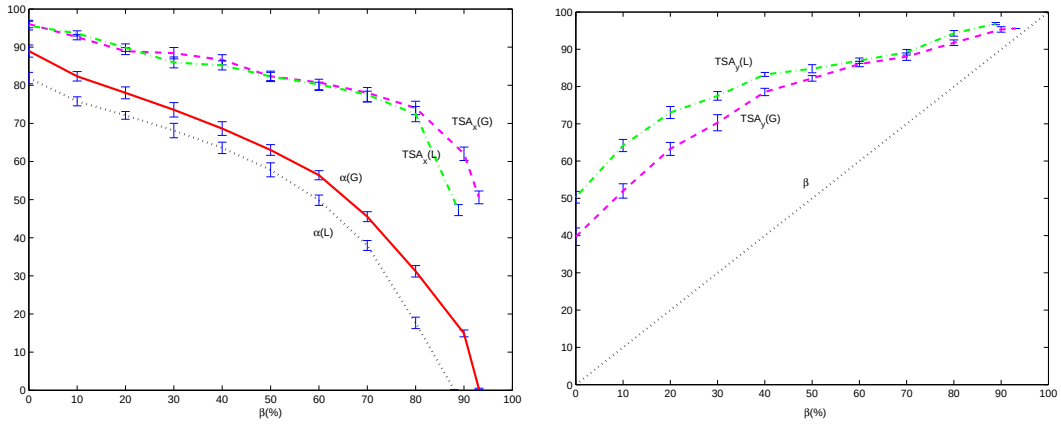
We begin with a lemma proposed in Popescu and Bertsimas (2001).

$$\sup_{\mathbf{y}\sim(\overline{\mathbf{y}},\Sigma_{\mathbf{y}})} \mathbf{Pr}\{\mathbf{y}\in\mathcal{S}\} = \frac{1}{1+d^2}, \quad \text{with} \quad d^2 = \inf_{\mathbf{y}\in\mathcal{S}} (\mathbf{y}-\overline{\mathbf{y}})^T\Sigma_{\mathbf{y}}^{-1}(\mathbf{y}-\overline{\mathbf{y}}), \tag{24}$$

where $\mathcal{S}$ denotes a convex set.

If we define $\mathcal{S} = \{\mathbf{a}^T\mathbf{y} \geq b\}$, the above lemma is changed to:

$$\sup_{\mathbf{y}\sim\{\overline{\mathbf{y}},\Sigma_{\mathbf{y}}\}} \mathbf{Pr}\{\mathbf{a}^T\mathbf{y}\geq b\} = \frac{1}{1+d^2}, \quad \text{with} \quad d^2 = \inf_{\mathbf{a}^T\mathbf{y}\geq\mathbf{b}} (\mathbf{y}-\overline{\mathbf{y}})^T\Sigma_{\mathbf{y}}^{-1}(\mathbf{y}-\overline{\mathbf{y}}) .$$



(a) (b)

Figure 7: Bounds and real accuracies. With β_0 varying from 0 to 1, the real accuracies are lower bounded by the worst-case accuracies. In addition, α(G) is above α(L), which shows the power of the kernelization.

By reference to (3), for a given hyperplane $\{\mathbf{a}, b\}$, we can easily obtain that

$$\beta = \frac{d^2}{1+d^2} \,. \tag{25}$$

Moreover, in Lanckriet et al. (2002b), a simple closed-form expression for the minimum distance d is derived:

$$d^2 = \inf_{\mathbf{a}^T\mathbf{y} \geq b} (\mathbf{y}-\overline{\mathbf{y}})^T \Sigma_{\mathbf{y}}^{-1} (\mathbf{y}-\overline{\mathbf{y}}) = \frac{\max((b-\mathbf{a}^T\overline{\mathbf{y}}),0)^2}{\mathbf{a}^T\Sigma_{\mathbf{y}}\mathbf{a}} \,. \tag{26}$$

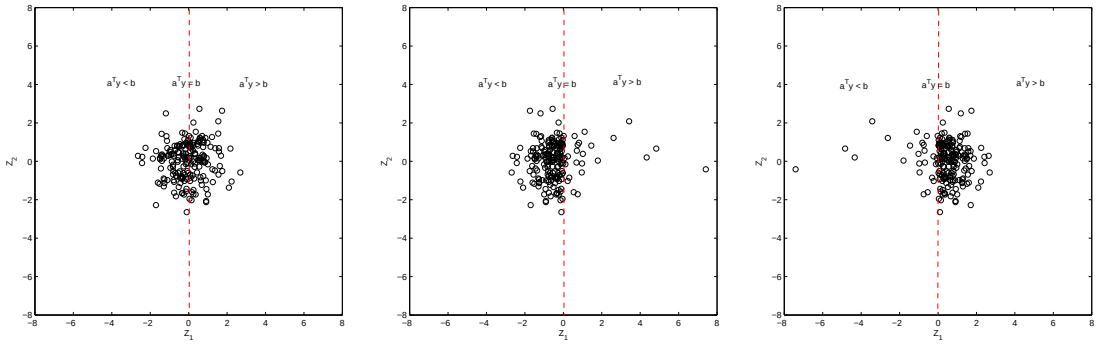
It is easy to see that when the decision hyperplane $\{\mathbf{a}, b\}$ passes the center $\overline{\mathbf{y}}$, d would be equal to 0 and the worst-case accuracy β would be 0 according to (25).

However, if we consider the Gaussian data (which we assume as $\mathbf{y}$ data) in Figure 8(a), a vertical line approximating $\overline{\mathbf{y}}$ would achieve about 50% test accuracy. The large gap between the worst-case accuracy and the real test accuracy seems strange. In the following, we construct an example of one-dimensional data to show the inner rationality of this observation. We attempt to provide the worst-case distribution containing the given mean and covariance, while a hyperplane passing its mean achieves a real test accuracy of zero.

Consider one-dimensional data y consisting of $N-1$ observations with values as m and one single observation with the value as $\sigma\sqrt{N}+m$. If we calculate the mean and the covariance, we obtain:

$$\overline{y} = m + \frac{\sigma}{\sqrt{N}} \,,$$
$$\Sigma_y = \frac{N-1}{N}\sigma^2 \,.$$

When N goes to infinity, the above one-dimensional data have the mean as m and the covariance as σ. In this extreme case, a hyperplane passing the mean will achieve a zero test accuracy, which is exactly the worst-case accuracy given the fixed mean and covariance as m and σ respectively. This example demonstrates the inner rationality of the minimax probability machines.



(a) (b) (c)

Figure 8: Three two-dimensional data sets with the same means and covariances but with different skewness. The worst-case accuracy bound of (a) is tighter than that of (b) and looser than that of (c).

To further examine the tightness of the worst-case bound in Figure 8(a), we vary β from 0 to 1 and plot against β the real test accuracy that a vertical line classifies the $\mathbf{y}$ data by using (25). Note that the real accuracy can be calculated as $\Phi(z \leq d)$. This curve is plotted in Figure 9.

Observed from Figure 9, the smaller the worst-case accuracy is, the looser it is. On the other hand, if we skew the $\mathbf{y}$ data towards the left side, while simultaneously maintaining the mean and covariance unchanged (see Figure 8(b)), an even bigger gap will be generated when β is small; similarly, if we skew the data towards the right side (see Figure 8(c)), a tighter accuracy bound will be expected. This finding means that adopting up to the second order moments only may not achieve a satisfactory bound. In other words, for a tighter bound, higher order moments such as skewness may need to be considered. This problem of estimating a probability bound based on moments is presented as the (n,k,Ω)-bound problem, which means "finding the tightest bound for an n-dimensional variable in the set Ω based on up to the k-th moments." Unfortunately, as proved in Popescu and Bertsimas (2001), it is NP-hard for $(n,k,\mathbb{R}^n)$-bound problems with $k \geq 3$. Thus tightening the bound by simply scaling up the moment order may be intractable in this sense. We may have to exploit other statistical techniques to achieve this goal. This certainly deserves a closer examination in the future.

7. On the Concavity of MEMPM

We address the issue of the concavity on the MEMPM model in this section. We will demonstrate that, although MEMPM cannot generally guarantee its concavity, there is strong empirical evidence showing that many real world problems demonstrate reasonable concavity in MEMPM. Hence, the

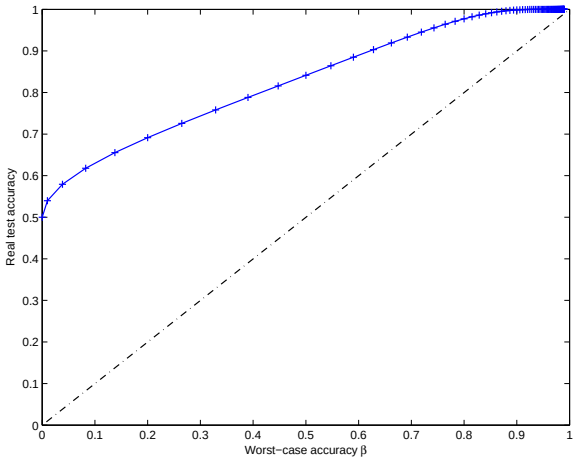


Figure 9: Theoretical comparison between the worst-case accuracy and the real test accuracy for the Gaussian data in Figure 8(a).

MEMPM model can be solved successfully by standard optimization methods, e.g., the linear search method proposed in this paper.

We first present a lemma for the BMPM model.

Lemma 8 *The optimal solution for BMPM is a strictly and monotonically decreasing function with respect to* β_0.

Proof Let the corresponding optimal worst-case accuracies on $\mathbf{x}$ be α_1 and α_2 respectively, when β_{01} and β_{02} are set to the acceptable accuracy levels for $\mathbf{y}$ in BMPM. We will prove that if $\beta_{01} > \beta_{02}$, then $\alpha_1 < \alpha_2$.

This can be proved by considering the contrary case, i.e., we assume $\alpha_1 \geq \alpha_2$. From the problem definition of BMPM, we have:

$$\begin{aligned} &\alpha_1 \geq \alpha_2 \Longrightarrow \kappa(\alpha_1) \geq \kappa(\alpha_2) \\ &\Longrightarrow \frac{1-\kappa(\beta_{01})\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{y}\mathbf{a_1}}}{\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{x}\mathbf{a_1}}} \geq \frac{1-\kappa(\beta_{02})\sqrt{\mathbf{a_2}^T\Sigma_\mathbf{y}\mathbf{a_2}}}{\sqrt{\mathbf{a_2}^T\Sigma_\mathbf{x}\mathbf{a_2}}}\,, \end{aligned} \tag{27}$$

where $\mathbf{a}_1$ and $\mathbf{a}_2$ are the corresponding optimal solutions that maximize $\kappa(\alpha_1)$ and $\kappa(\alpha_2)$ respectively, when β_{01} and β_{02} are specified.

From $\beta_{01} > \beta_{02}$ and (27), we have

$$\frac{1-\kappa(\beta_{02})\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{y}\mathbf{a_1}}}{\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{x}\mathbf{a_1}}} > \frac{1-\kappa(\beta_{01})\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{y}\mathbf{a_1}}}{\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{x}\mathbf{a_1}}} \geq \frac{1-\kappa(\beta_{02})\sqrt{\mathbf{a_2}^T\Sigma_\mathbf{y}\mathbf{a_2}}}{\sqrt{\mathbf{a_2}^T\Sigma_\mathbf{x}\mathbf{a_2}}}\,. \tag{28}$$

On the other hand, since $\mathbf{a}_2$ is the optimal solution of $\max_\mathbf{a} \frac{1-\kappa(\beta_{02})\sqrt{\mathbf{a}^T\Sigma_\mathbf{y}\mathbf{a}}}{\sqrt{\mathbf{a}^T\Sigma_\mathbf{x}\mathbf{a}}}$, we have:

$$\frac{1-\kappa(\beta_{02})\sqrt{\mathbf{a_2}^T\Sigma_\mathbf{y}\mathbf{a_2}}}{\sqrt{\mathbf{a_2}^T\Sigma_\mathbf{x}\mathbf{a_2}}} \geq \frac{1-\kappa(\beta_{02})\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{y}\mathbf{a_1}}}{\sqrt{\mathbf{a_1}^T\Sigma_\mathbf{x}\mathbf{a_1}}}\,.$$

This is obviously contradictory to (28). ■

From the sequential solving method of MEMPM, we know that MEMPM actually corresponds to a one-dimensional line search problem. More specifically, it further corresponds to maximizing the sum of two functions, namely, $f_1(\beta) + f_2(\beta)$,[4] where $f_1(\beta)$ is determined by the BMPM optimization and $f_2(\beta) = \beta$. According to Lemma 8, $f_1(\beta)$ strictly decreases as β increases. Thus it is strictly pseudo-concave. However, generally speaking, the sum of a pseudo-concave function and a linear function is not necessarily a pseudo-concave function and thus we cannot assure that every local optimum is the global optimum. This can be clearly observed in Figure 10. In this figure, f_1 is pseudo-concave in all three sub-figures; however, the sum $f_1 + f_2$ does not necessarily lead to a pseudo-concave function.

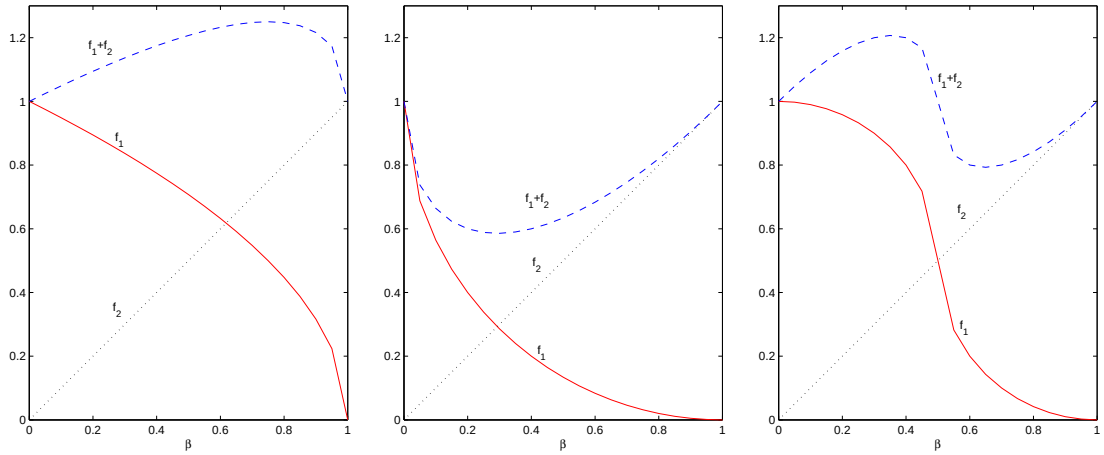
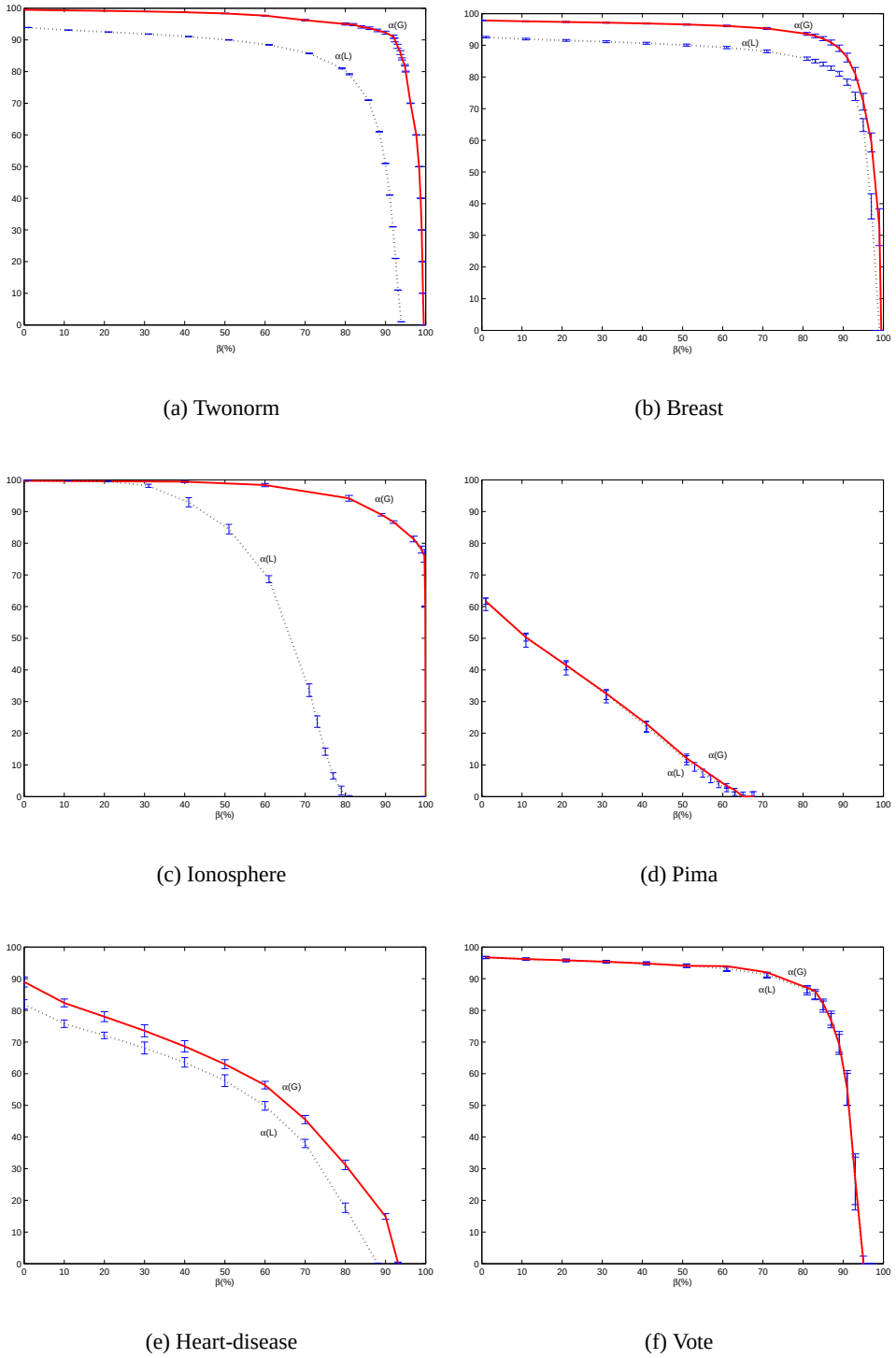


Figure 10: The sum of a pseudo-concave function and a linear function is not necessarily a concave function. In (a), $f_1 + f_2$ is a concave function, however in (b) and (c) it is not.

Nevertheless, there is strong empirical evidence showing that for many "well-behaved" real world classification problems, f_1 is overall concave, which results in the concavity of $f_1 + f_2$. This is first verified by the data sets used in this paper. We shift β from 0 to the corresponding upper bound and plot α against β in Figure 11. It is clearly observed that in all six data sets including both kernel and linear cases, the curves of α against β are overall concave. This motivates us to look further into the concavity of MEMPM. As shown in the following, when two classes of data are "well-separated," f_1 would be concave in the main "interest" region.

We analyze the concavity of $f_1(\beta)$ by imagining that β changes from 0 to 1. In this process, the decision hyperplane moves slowly from $\overline{\mathbf{y}}$ to $\overline{\mathbf{x}}$ according to (25) and (26). At the same time, $\alpha = f_1(\beta)$ should decrease accordingly. More precisely, if we denote $d_{\mathbf{x}}$ and $d_{\mathbf{y}}$ respectively as the

4. For simplicity, we assume θ as 0.5. Since a constant does not influence the concavity analysis, the factor of 0.5 is simply dropped.



(a) Twonorm (b) Breast

(c) Ionosphere (d) Pima

(e) Heart-disease (f) Vote

Figure 11: The curves of α against β (f_1) all tend to be concave in the data sets used in this paper.

Mahalanobis distances that $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are from the associated decision hyperplane with a specified β, we can formulate the changing of α and β as

$$\alpha \to \alpha - k_1(d_{\mathbf{x}})\Delta d_{\mathbf{x}},$$
$$\beta \to \beta + k_2(d_{\mathbf{y}})\Delta d_{\mathbf{y}},$$

where $k_1(d_{\mathbf{x}})$ and $k_2(d_{\mathbf{y}})$ can be considered as the changing rate of α and β when the hyperplane lies $d_{\mathbf{x}}$ distance far away from $\overline{\mathbf{x}}$ and $d_{\mathbf{y}}$ distance far away from $\overline{\mathbf{y}}$ respectively. We consider the changing of α against β, namely, f_1':

$$f_1' = \frac{-k_1(d_{\mathbf{x}})\Delta d_{\mathbf{x}}}{k_2(d_{\mathbf{y}})\Delta d_{\mathbf{y}}}.$$

If we consider $d_{\mathbf{x}}$ and $\Delta d_{\mathbf{y}}$ insensitively change against each other or change with a proportional rate, i.e., $\Delta d_{\mathbf{x}} \approx c\Delta d_{\mathbf{y}}$ (c is a positive constant) as the decision hyperplane moves, the above equation can further be written as $f_1' = c\frac{-k_1(d_{\mathbf{x}})}{k_2(d_{\mathbf{y}})}$.

Lemma 9 *(1) If $d_{\mathbf{y}} \geq 1/\sqrt{3}$ or the corresponding $\beta \geq 0.25$, $k_2(d_{\mathbf{y}})$ decreases as $d_{\mathbf{y}}$ increases.*
(2) If $d_{\mathbf{x}} \geq 1/\sqrt{3}$ or the corresponding $\alpha \geq 0.25$, $k_1(d_{\mathbf{x}})$ decreases as $d_{\mathbf{x}}$ increases.

Proof Since (1) and (2) are very similar statements, we only prove (1). Note that $k_2(d)$ is the first order derivative of $\frac{d^2}{1+d^2}$ according to (25). We consider the first order derivative of $k_2(d)$ or the second order derivative of $\frac{d^2}{1+d^2}$. It is easily verified that $(\frac{d^2}{1+d^2})'' \leq 0$ when $d \geq 1/\sqrt{3}$. This is also illustrated in Figure 12. According to the definition of the second order derivative, we immediately obtain the lemma. Note that $d \geq 1/\sqrt{3}$ corresponds to $\beta \geq 0.25$. Thus the condition can also be replaced by $\beta \geq 0.25$.

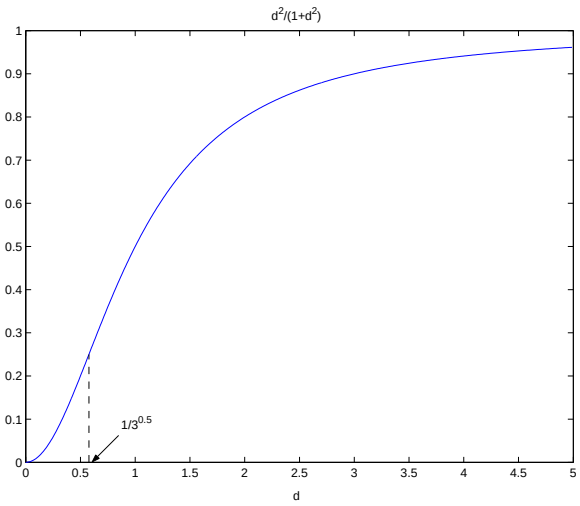


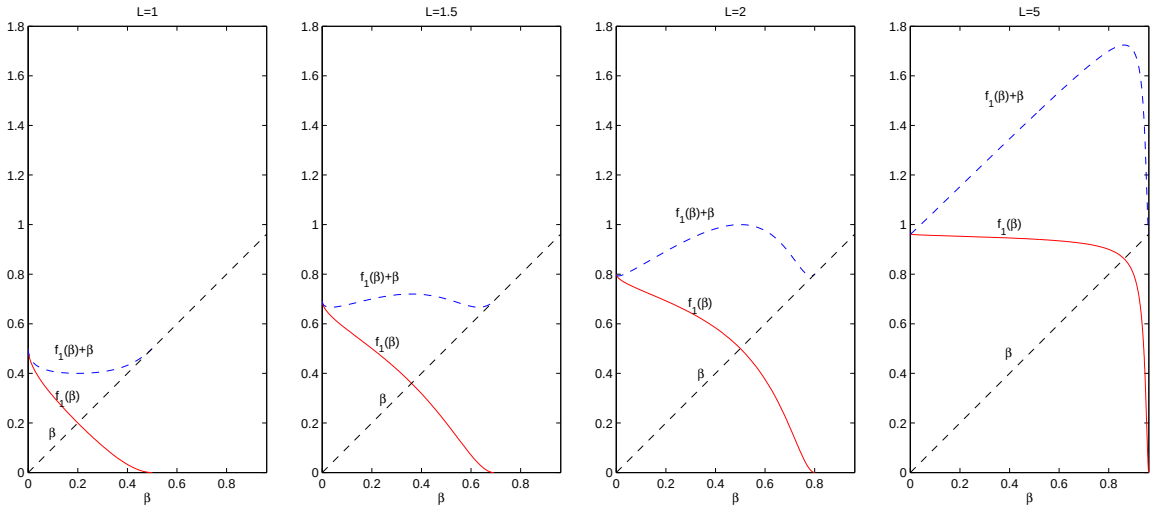
Figure 12: The curve of $d^2/(1+d^2)$. This function is concave when $d \geq 1/\sqrt{3}$.

■

In the above procedure, $d_{\mathbf{y}}$, β increase and $d_{\mathbf{x}}$, α decrease, as the hyperplane moves towards $\overline{\mathbf{x}}$.

Therefore, according to Lemma 9, $k_1(d_{\mathbf{x}})$ increases while $k_2(d_{\mathbf{y}})$ decreases when $\alpha, \beta \in [0.25,\ 1)$. This shows that f_1' is getting smaller as the hyperplane moves towards $\overline{\mathbf{x}}$. In other words, f_1'' would be less than 0, and it is concave when $\alpha, \beta \in [0.25,\ 1)$. It should be noted that in many well-separated real world data sets, there is a high possibility that the optimal α and β will be greater than 0.25, since to achieve good performance, the worst-case accuracies are naturally required to be greater than a certain small amount, e.g., 0.25. This is observed in the data sets used in the paper. All the data sets except the Pima data attain their optima satisfying this constraint. For Pima, the overall accuracy is relatively lower, which implies two classes of data in this data set appear to overlap substantially with each other.[5]

An illustration can also be seen in Figure 13. We generate two classes of Gaussian data with $\overline{\mathbf{x}} = [0,\ 0]^T$, $\overline{\mathbf{y}} = [L,\ 0]^T$, and $\Sigma_{\mathbf{x}} = \Sigma_{\mathbf{y}} = [1,\ 0; 0,\ 1]$. The prior probability for each data class is set to an equal value 0.5. We plot the curves of $f_1(\beta)$ and $f_1(\beta) + \beta$ when L is set to different values. It is observed that when two classes of data substantially overlap with each other, for example in Figure 13(a) with $L = 1$, the optimal solution of MEMPM lies in the small-value range of α and β, which is usually not concave. On the other hand, (b), (c), and (d) show that when two classes of data are well-separated, the optimal solutions lie in the region with $\alpha, \beta \in [0.25,\ 1)$, which is often concave.



(a) (b) (c) (d)

Figure 13: An illustration of the concavity of the MEMPM. Subfigure (a) shows that when two classes of data overlap substantially with each other, the optimal solution of MEMPM lies in the small-value range of α and β, which is usually not concave. (b), (c), and (d) show that when two classes of data are well-separated, the optimal solutions lie in the region with $\alpha, \beta \in [0.25,\ 1)$, which is often concave.

5. It is observed, even for Pima, the proposed solving algorithm is still successful, since α is approximately linear as shown in Figure 11. Moreover, due to the fact that the slope of α is slightly greater than -1, the final optimum naturally leads β to achieve its maximum.

Note that, in the above, we make an assumption that as the decision hyperplane moves, $d_{\mathbf{x}}$ and $d_{\mathbf{y}}$ change at an approximately fixed proportional rate. From the definition of $d_{\mathbf{x}}$ and $d_{\mathbf{y}}$, this assumption implies that $\mathbf{a}$, the direction of the optimal decision hyperplane, is insensitive to β. This assumption does not hold in all cases; however, observed from the geometrical interpretation of MEMPM, for those data with isotropic or not significantly anisotropic $\Sigma_{\mathbf{x}}$ and $\Sigma_{\mathbf{y}}$, $\mathbf{a}$ would indeed be insensitive to β.

We summarize the above analysis in the following proposition.

Proposition 10 *Assuming (1) two classes of data are well-separated and (2) $d_{\mathbf{x}}$ and $d_{\mathbf{y}}$ change at an approximately fixed proportional rate as the optimal decision hyperplane (associated with a specified β) moves, the one-dimensional line search problem of MEMPM is often concave in the range of $\alpha, \beta \in [0.25, 1)$ and will often attain its optimum in this range. Therefore the proposed solving method leads to a satisfactory solution.*

Remarks. As demonstrated in the above, although the MEMPM is often overall concave in real world tasks, there exist cases that the MEMPM optimization problem is not concave. This may lead to a local optimum, which may not be the global optimum. In this case, we may need to choose the initial starting point carefully. In addition, the physical interpretation of β as the worst-case accuracy may make it relatively easy to choose a suitable initial value. For example, we can set the initial value by using the information obtained from prior domain knowledge.

8. Limitations and Future Work

In this section, we present the limitations and future work. First, although MEMPM achieves better performance than the MPM, its sequential optimization of the Biased Minimax Probability Machine may cost more training time than MPM. Although in pattern recognition tasks, especially in off-line classification, effectiveness is often more important than efficiency, expensive time-cost presents one of the main limitations of the MEMPM model, in particular for large scale data sets with millions of samples. To solve this problem, one possible direction is to eliminate those redundant points that make less contribution to the classification. In this way, the problem dimension (in the kernelization) would be greatly decreased and this may help in reducing the computational time required. Another possible direction is to exploit some techniques to decompose the Gram matrix (as is done in SVM) and to develop some specialized optimization procedures for MEMPM. Recently, we also note that Strohmann et al. (2004) have proposed a speed-up method by exploiting the sparsity of MPM. Undoubtedly, speeding up the algorithm will be a highly worthy topic in the future.

Second, as a generalized model, MEMPM actually incorporates some other variations. For example, when the prior probability (θ) cannot be estimated reliably (e.g., in sparse data), maximizing $\alpha + \beta$, namely the sum of the accuracies or the difference between true positive and false positive, would be considered. This scheme is widely used in the pattern recognition field, e.g., in medical diagnosis (Grzymala-Busse et al., 2003) and in graph detection, especially line detection and arc detection, where it is called the Vector Recovery Index (Liu and Dori, 1997; Dori and Liu, 1999). Moreover, when there are domain experts at hand, a variation of MEMPM, namely, the maximization of $C_{\mathbf{x}}\alpha + C_{\mathbf{y}}\beta$ may be used, where $C_{\mathbf{x}}$ ($C_{\mathbf{y}}$) is the cost of a misclassification of $\mathbf{x}$ ($\mathbf{y}$) obtained from experts. Exploring these variations in some specific domains is thus a valuable direction in the future.

Third, we have proposed a general framework for robustly estimating model input parameters, namely, the means and covariances. Based on this framework, estimating the input vector or matrix parameters is changed to finding four adapting scale parameters, i.e., $\nu_{\mathbf{x}}, \nu_{\mathbf{y}}, \rho_{\mathbf{x}}$, and $\rho_{\mathbf{y}}$. While we may obtain these four parameters by conducting cross validation in small data sets, it is computationally hard to do this in large scale data sets. Although one possible way to determine these values is based on the central limit theorem or the resampling method (Lanckriet et al., 2002b), it is still valuable to investigate other techniques in the future.

Fourth, Lanckriet et al. (2002b) have built up a connection between MPM and SVM from the perspective of the margin definition, i.e., MPM corresponds to finding the hyperplane with the maximal margin from the class center. Nevertheless, some deeper connections need to be investigated, e.g., how is the bound of MEMPM related to the generalization bound of SVM? More recently, Huang et al. (2004a) have disclosed the relationship between them from either a local or a global viewpoint of data. It is particularly useful to look into these links and explore their further connections in the future.

9. Conclusion

The Minimax Probability Machine achieves performance in classification tasks that is comparable to that of a state-of-the-art classifier, the Support Vector Machine. This model attempts to minimize the worst-case probability of misclassification of future data points. However, its equality constraint on the worst-case accuracies for two classes makes it unnecessarily minimize the error rate in the worst-case setting and thus cannot assure the optimal classifier in this sense.

In this paper, we have proposed a generalized Minimax Probability Machine, called the Minimum Error Minimax Probability Machine, which removes the equality constraint on the worst-case accuracies for two classes. By minimizing the upper bound of the Bayes error of future data points, our approach derives the distribution-free Bayes optimal hyperplane in the worst-case setting. More importantly, we have shown that the worst-case Bayes optimal hyperplane derived by MEMPM becomes the true Bayes optimal hyperplane when certain conditions are satisfied, in particular, when Gaussianity is assumed for the data. We have evaluated our algorithm on both synthetic data sets and real world benchmark data sets. The performance of MEMPM is demonstrated to be very promising. Moreover, the validity of our proposition, i.e., the minimum error rate Minimax Probability Machine is not certain to achieve the same worst-case accuracies for two classes, has also been verified by the experimental results.

Acknowledgements

We thank Gert R. G. Lanckriet at U.C. Berkeley for providing the Matlab source code of Minimax Probability Machine on the web. We extend our thanks to the editor and the anonymous reviewers for their valuable comments. The work described in this paper was fully supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4182/03E and Project No. CUHK4235/04E).

References

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 2nd edition, 1999.

C. L. Blake and C. J. Merz. Repository of machine learning databases, University of California, Irvine, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

L. Breiman. Arcing classifiers. Technical Report 460, Statistics Department, University of California, 1997.

Y. S. Chow and H. Teicher. *Probability theory: Independence, interchageability, martingales*. Springer-Verlag, New York, 3rd edition, 1997.

B. D. Craven. *Mathematical Programming and Control Theory*. Chapman and Hall, London, 1978.

B. D. Craven. *Fractional Programming, Sigma Series in Applied Mathematics 4*. Heldermann Verlag, Berlin, 1988.

G. Deco and D. Obradovic. *An information-theoretic approach to neural computing*. Springer-Verlag, 1996.

D. Dori and W. Liu. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:202–215, 1999.

J. W. Grzymala-Busse, L. K. Goodwin, and X. Zhang. Increasing sensitivity of preterm birth by changing rule strengths. *Pattern Recognition Letters*, 24:903–910, 2003.

K. Huang, H. Yang, I. King, and M. R. Lyu. Learning large margin classifiers locally and globally. In *The Twenty-First International Conference on Machine Learning (ICML-2004)*, pages 401–408, 2004a.

K. Huang, H. Yang, I. King, M. R. Lyu, and L. Chan. Biased minimax probability machine for medical diagnosis. In *the Eighth International Symposium on Artificial Intelligence and Mathematics (AMAI-2004)*, 2004b.

D. Keysers, F. J. Och, and H. Ney. Maximum entropy and gaussian models for image object recognition. In *Proceedings of the Twenty-Fourth DAGM Symposium, Volume 2449 of Lecture Notes in Computer Science*, pages 498–506. Springer-Verlag, 2002.

G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. Minimax probability machine. In *Advances in Neural Information Processing Systems (NIPS 14)*, 2002a.

G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002b.

W. Liu and D. Dori. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Application*, 9:240–250, 1997.

M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

M. A. Maloof, P. Langley, T. O. Binford, R. Nevatia, and S. Sage. Improved rooftop detection in aerial images with machine learning. *Machine Learning*, 53:157–191, 2003.

Y. Nesterov and A. Nemirovsky. *Interior point polynomial methods in convex programming: Theory and applications*. Studies in Applied Mathematics 13. Society for Industrial & Applied Math, 1994.

E. Osuna, R. Freund, and F. Girosi. Support Vector Machines: Training and Applications. Technical Report AIM-1602, MIT, 1997. URL `citeseer.nj.nec.com/osuna97support.html`.

I. Popescu and D. Bertsimas. Optimal inequalities in probability theory: A convex optimization approach. Technical Report TM62, INSEAD, 2001.

S. Schaible. Fractional programming. *Zeitschrift für Operational Research*, Serie A 27(1):39–54, 1977.

S. Schaible. Fractional programming. In R. Horst and P. M. Pardalos, editors, *Handbook of Global Optimization*, Nonconvex Optimization and its Applications, pages 495–608. Kluwer Academic Publishers, Dordrecht-Boston-London, 1995.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

T. Strohmann, A. Belitski, G. Grudic, and D. DeCoste. Sparse greedy minimax probability machine classification. In *Advances in Neural Information Processing Systems (NIPS 16)*. 2004.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 2nd edition, 1999.

Journal of Machine Learning Research 5 (2004) 1287–1330 Submitted 3/04; Published 10/04

Large-Sample Learning of Bayesian Networks is NP-Hard

David Maxwell Chickering DMAX@MICROSOFT.COM
David Heckerman HECKERMA@MICROSOFT.COM
Christopher Meek MEEK@MICROSOFT.COM
Microsoft Research
Redmond, WA 98052, USA

Editor: David Madigan

Abstract

In this paper, we provide new complexity results for algorithms that learn discrete-variable Bayesian networks from data. Our results apply whenever the learning algorithm uses a scoring criterion that favors the simplest structure for which the model is able to represent the generative distribution exactly. Our results therefore hold whenever the learning algorithm uses a consistent scoring criterion and is applied to a sufficiently large dataset. We show that identifying high-scoring structures is NP-hard, even when any combination of one or more of the following hold: the generative distribution is perfect with respect to some DAG containing hidden variables; we are given an independence oracle; we are given an inference oracle; we are given an information oracle; we restrict potential solutions to structures in which each node has at most k parents, for all $k \geq 3$.

Our proof relies on a new technical result that we establish in the appendices. In particular, we provide a method for constructing the local distributions in a Bayesian network such that the resulting joint distribution is provably perfect with respect to the structure of the network.

Keywords: learning Bayesian networks, search complexity, large-sample data, NP-Hard

1. Introduction

Researchers in the machine-learning community have generally accepted that without restrictive assumptions, learning Bayesian networks from data is NP-hard, and consequently a large amount of work in this community has been dedicated to heuristic-search techniques to identify good models. A number of discouraging complexity results have emerged over the last few years that indicate that this belief is well founded. Chickering (1996) shows that for a general and widely used class of Bayesian scoring criteria, identifying the highest-scoring structure from small-sample data is hard, even when each node has at most two parents. Dasgupta (1999) shows that it is hard to find the polytree with highest maximum-likelihood score. Although we can identify the highest-scoring tree structure using a polynomial number of calls to the scoring criterion, Meek (2001) shows that identifying the best *path structure*—that is, a tree in which each node has degree at most two—is hard. Bouckaert (1995) shows that for domains containing only binary variables, finding the parameter-minimal structure that is consistent with an independence oracle is hard; we discuss this result in more detail below. Finally, Srebro (2001) shows that it is hard to find Markov networks with bounded tree width that maximize the maximum-likelihood score.

In this paper, we are interested in the large-sample version of the learning problem considered by Chickering (1996). The approach used by Chickering (1996) to reduce a known NP-complete problem to the problem of learning is to construct a complicated prior network that defines the

Bayesian score, and then create a dataset consisting of a single record. Although the result is discouraging, the proof technique leaves open the hope that, in scenarios where the network scores are more "well behaved", learning is much easier.

As the number of records in the observed data grows large, most scoring criteria will agree on the same partial ranking of model structures; in particular, any *consistent* scoring criterion will—in the limit of large data—favor a structure that allows the model to represent the generative distribution over a structure that does not, and when comparing two structures that both allow the model to represent the generative distribution, will favor the structure that results in fewer model parameters. Almost all scoring criteria used in practice are consistent, including (1) any Bayesian criterion that does not rule out model structures apriori, (2) the minimum-description-length criterion, and (3) the Bayesian-information criterion.

In this paper, we consider the scenario when a learning algorithm is using a consistent scoring criterion with a large dataset. We assume that the learning algorithm has direct access to the generative distribution itself; the resulting learning problem is to identify the simplest DAG that allows the resulting Bayesian network to represent the generative distribution exactly. There are a number of algorithms that have been developed for this large-sample learning problem. The SGS algorithm (Spirtes, Glymour and Scheines, 2000), the GES algorithm (Meek, 1997; Chickering, 2002), and the KES algorithm (Nielsen, Kočka and Peña, 2003) all identify the optimal DAG if there exists a solution in which all independence and dependence relationships implied by that structure hold in the generative distribution (that is, the generative distribution is *DAG perfect* with respect to the observable variables). Unfortunately, none of these algorithms run in polynomial time in the worst case.

With some restrictive assumptions, however, we can accomplish large-sample learning efficiently. If (1) the generative distribution is DAG perfect with respect to the observable variables and (2) we know that there exists a solution in which each node has at most k parents (for some constant k), then we can apply the SGS algorithm to identify the best network structure in a polynomial number of independence tests. In particular, because we know the value k, we can limit the worst-case number of independence tests used by the algorithm. Alternatively, if (1) the generative distribution is DAG perfect with respect to *some* DAG that might contain vertices corresponding to hidden variables, and (2) we are given a total ordering over the variables that is consistent with the best structure, then we can find the best DAG using a polynomial number of calls to the scoring criterion by applying a version of the GES algorithm that greedily adds and then deletes the parents of each node.

Unfortunately, the assumptions needed for these special-case efficient solutions are not likely to hold in most real-world scenarios. In this paper, we show that in general—without the assumption that the generative distribution is DAG perfect with respect to the observables and without the assumption that we are given a total ordering—large-sample learning is NP-hard. We demonstrate that learning is NP-hard even when (1) the generative distribution is perfect with respect to a DAG (which contains hidden variables), (2) we are given an independence oracle, (3) we are given given an inference oracle, and/or (4) we are given an information oracle. We show that these results also apply to the problem of identifying high-scoring structures in which each node has at most k parents, for all $k \geq 3$.

A secondary contribution of this paper is a general result about Bayesian networks: in the appendices of this paper, we identify two properties of the local distributions in a Bayesian network

that are sufficient to guarantee that all independence and dependence facts implied by the structure also hold in the joint distribution. Our NP-hard proof relies on this result.

As an extension of our main result, we consider the case in which we are given an independence oracle, and we show in Theorem 15 that the resulting learning problem remains NP-hard. This theorem extends the independence-oracle result of Bouckaert (1995) in a number of ways. Perhaps most important, we place no restriction on the number of states for the (discrete) variables in the domain, which proves the conjecture in Bouckaert (1995) that learning with an independence oracle in non-binary domains is NP-hard. Another extension we make has to do with assumptions about the generative distribution. In the elegant reduction proof of Bouckaert (1995), the constructed independence oracle is consistent with a particular generative distribution that is not perfect with respect to any DAG. Although this distribution has properties that yield a much simpler reduction than our own, the results of this paper apply under the common assumption in the machine-learning literature that the generative distribution is, in fact, perfect with respect to some DAG. Furthermore, the DAG we use in the reduction, which contains hidden variables, has a sparse dependency structure: each node has at most two parents. Finally, our result extends the Bouckaert (1995) oracle-learning result to scenarios where we want to identify sparse (i.e., parent-count limited) model structures that are consistent with an oracle.

2. Background

In this section, we provide background material relevant to the rest of the paper. We denote a variable by an upper case token (e.g., A, B_i, Y) and a state or value of that variable by the same token in lower case (e.g., a, b_i, y). We denote sets with bold-face capitalized tokens (e.g., $\mathbf{A}, \mathbf{B}$) and corresponding sets of values by bold-face lower case tokens (e.g., $\mathbf{a}, \mathbf{b}$). Finally, we use calligraphic tokens (e.g., $\mathcal{B}, \mathcal{G}$) to denote Bayesian networks and graphs.

In this paper, we concentrate on Bayesian networks for a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, where each $X_i \in \mathbf{X}$ has a finite number of states. A *Bayesian network* for a set of variables $\mathbf{X}$ is a pair $(\mathcal{G}, \theta_{\mathcal{G}})$ that defines a joint probability distribution over $\mathbf{X}$. $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is an acyclic directed graph—or *DAG* for short—consisting of (1) nodes $\mathbf{V}$ in one-to-one correspondence with the variables $\mathbf{X}$, and (2) directed edges $\mathbf{E}$ that connect the nodes. $\theta_{\mathcal{G}}$ is a set of parameter values that specify the conditional probability distributions that collectively define the joint distribution.

We assume that each conditional probability distribution is a full table. That is, for each variable there is a separate (unconstrained) multinomial distribution given every distinct configuration of the parent values. For a variable X_i with r_i states, $r_i - 1$ parameters are both necessary and sufficient to specify an arbitrary multinomial distribution over X_i. Thus, assuming that there are q_i distinct parent configurations for X_i, the conditional distribution for X_i will contain $(r-1) \cdot q_i$ parameter values. We also assume that the number of states for each variable is some constant that does not depend on the number of variables in the domain.

Learning a Bayesian network from data requires both identifying the model structure $\mathcal{G}$ and identifying the corresponding set of model parameter values $\theta_{\mathcal{G}}$. Given a fixed structure, however, it is straightforward to estimate the parameter values. As a result, research on the problem of learning Bayesian networks from data is focused on methods for identifying one or more "good" DAG structures from data.

All independence constraints that necessarily hold in the joint distribution represented by any Bayesian network with structure $\mathcal{G}$ can be identified by the by the *d-separation* criterion of Pearl

(1988) applied to to $\mathcal{G}$. In particular, two nodes X and Y are said to be d-separated in a DAG $\mathcal{G}$ given a set of nodes $\mathbf{O}$ if and only if there is no $\mathbf{O}$-*active path* in $\mathcal{G}$ between X and Y; an $\mathbf{O}$-active path is a simple path for which each node Z along the path either (1) has converging arrows and Z or a descendant of Z is in $\mathbf{O}$ or (2) does not have converging arrows and Z is not in $\mathbf{O}$. By simple, we mean that the path never passes through the same node twice. If two nodes are not d-separated given some set, we say that they are *d-connected* given that set. We use $X \perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$ to denote the assertion that DAG $\mathcal{G}$ imposes the constraint—via d-separation—that for all values $\mathbf{z}$ of the set $\mathbf{Z}$, X is independent of Y given $\mathbf{Z} = \mathbf{z}$. For a probability distribution $p(\cdot)$, we use $X \perp\!\!\!\perp_p Y|\mathbf{Z}$ to denote the assertion that for all values $\mathbf{z}$ of the set $\mathbf{Z}$, X is independent of Y given $\mathbf{Z} = \mathbf{z}$ in p.

We say that a distribution $p(\mathbf{X})$ is *Markov* with respect to a DAG $\mathcal{G}$ if $X \perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$ implies $X \perp\!\!\!\perp_p Y|\mathbf{Z}$. Similarly, we say that $p(\mathbf{X})$ is *faithful* with respect to $\mathcal{G}$ if $X \perp\!\!\!\perp_p Y|\mathbf{Z}$ implies $X \perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$. If p is both Markov and faithful with respect to $\mathcal{G}$, we say that p is *perfect* with respect to $\mathcal{G}$. Note that if p is faithful with respect to $\mathcal{G}$, then $X \not\perp\!\!\!\perp_{\mathcal{G}} Y|\mathbf{Z}$ implies that *there exists* some x, y and $\mathbf{z}$ such that $p(x,y|\mathbf{z}) \neq p(x|\mathbf{z})p(y|\mathbf{z})$; there may be other values for which equality holds. We say that $p(\mathbf{X})$ is *DAG perfect* if there exists a DAG $\mathcal{G}$ such that $p(\mathbf{X})$ is perfect with respect to $\mathcal{G}$.

We say that a DAG $\mathcal{G}$ *includes* a distribution $p(\mathbf{X})$—and that $p(\mathbf{X})$ is *included by* $\mathcal{G}$—if the distribution can be represented by some Bayesian network with structure $\mathcal{G}$. Because we are only considering Bayesian networks that have complete tables as conditional distributions, $\mathcal{G}$ includes $p(\mathbf{X})$ if and only if $p(\mathbf{X})$ is Markov with respect to $\mathcal{G}$. We say that two DAGs $\mathcal{G}$ and $\mathcal{G}'$ are *equivalent* if the two sets of distributions included by $\mathcal{G}$ and $\mathcal{G}'$ are the same. Due to the complete-table assumption, an equivalent definition is that $\mathcal{G}$ and $\mathcal{G}'$ are equivalent if they impose the same independence constraints (via d-separation). For any DAG $\mathcal{G}$, we say an edge $X \rightarrow Y$ is *covered* in $\mathcal{G}$ if X and Y have identical parents, with the exception that X is not a parent of itself. The significance of covered edges is evident from the following result:

Lemma 1 (Chickering, 1995) *Let $\mathcal{G}$ be any DAG, and let $\mathcal{G}'$ be the result of reversing the edge $X \rightarrow Y$ in $\mathcal{G}$. Then $\mathcal{G}'$ is a DAG that is equivalent to $\mathcal{G}$ if and only if $X \rightarrow Y$ is covered in $\mathcal{G}$.*

As described above, when a Bayesian network has complete tables, the number of parameters is completely determined by its DAG and the number of states for each variable in the domain. To simplify presentation, we assume that the number of states for the variable corresponding to each vertex in a DAG is available implicitly, and therefore we can define the number of parameters associated with a DAG without reference to the corresponding state counts. In particular, we say that a DAG *supports* a number of parameters k when all Bayesian networks with that structure (defined over a particular domain) contain k parameters. The following result follows immediately from Lemma 1 for Bayesian networks with complete tables:

Lemma 2 (Chickering, 1995) *If $\mathcal{G}$ and $\mathcal{G}'$ are equivalent, then they support the same number of parameters.*

We say that DAG $\mathcal{H}$ *includes* DAG $\mathcal{G}$ if every distribution included by $\mathcal{G}$ is also included by $\mathcal{H}$. As above, an alternative but equivalent definition—due to the assumption of complete-table Bayesian networks—is that $\mathcal{H}$ includes $\mathcal{G}$ if every independence constraint implied by $\mathcal{H}$ is also implied by $\mathcal{G}$. Note that we are using "includes" to describe the relationship between a DAG and a particular distribution, as well as a relationship between two DAGs.

Theorem 3 (Chickering, 2002) *If $\mathcal{H}$ includes $\mathcal{G}$, then there exists a sequence of single edge additions and covered edge reversals in $\mathcal{G}$ such that (1) after each addition and reversal, $\mathcal{G}$ remains a DAG, (2) after each addition and reversal, $\mathcal{H}$ includes $\mathcal{G}$, and (3) after all additions and reversals, $\mathcal{G} = \mathcal{H}$.*

The "converse" of Theorem 3 will also prove useful.

Lemma 4 *If $\mathcal{F}$ can be transformed into $\mathcal{G}$ by a series of single edge additions and covered edge reversals, such that after each addition and reversal $\mathcal{F}$ remains a DAG, then $\mathcal{G}$ includes $\mathcal{F}$.*

Proof: Follows immediately from Lemma 1 and from the fact that the DAG that results from adding a single edge to $\mathcal{F}$ necessarily includes $\mathcal{F}$. □

3. Main Results

In this section, we provide the main results of this paper. We first define the decision problems that we use to prove that learning is NP-hard. As discussed in Section 1, in the limit of large data, all consistent scoring criteria rank network structures that include the generative distribution over those that do not, and among those structures that include the generative distribution, the criteria rank according to the number of parameters supported—with simpler structures receiving better scores. Thus, a natural decision problem corresponding to large-sample learning is the following:

LEARN
INSTANCE: Set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, probability distribution $p(\mathbf{X})$, and constant parameter bound d.
QUESTION: Does there exist a DAG that includes p and supports $\leq d$ parameters?

It is easy to see that if there exists an efficient algorithm for learning the optimal Bayesian-network structure from large-sample data, we can use that algorithm to solve LEARN: simply learn the best structure and evaluate the number of parameters it supports. By showing that LEARN is NP-hard, we therefore immediately conclude that the optimization problem of *identifying* the optimal DAG is hard as well. We show that LEARN is NP-hard using a reduction from a restricted version of the NP-complete problem FEEDBACK ARC SET. The general FEEDBACK ARC SET problem is stated by Garey and Johnson (1979) as follows:

FEEDBACK ARC SET
INSTANCE: Directed graph $\mathcal{G} = (\mathbf{V}, \mathbf{A})$, positive integer $k \leq |\mathbf{A}|$.
QUESTION: Is there a subset $\mathbf{A}' \subset \mathbf{A}$ with $|\mathbf{A}'| \leq k$ such that $\mathbf{A}'$ contains at least one arc from every directed cycle in $\mathcal{G}$?

Gavril (1977) shows that FEEDBACK ARC SET remains NP-complete for directed graphs in which no vertex has a total in-degree and out-degree more than three. We refer to this restricted version as DEGREE-BOUNDED FEEDBACK ARC SET, or *DBFAS* for short.

The remainder of this section is organized as follows. In Section 3.1, we describe a polynomial-time reduction from instances of DBFAS to instances of LEARN. In Section 3.2, we describe the main result of the appendices upon which Section 3.3 relies; in Section 3.3, we prove that there is a solution to an instance of DBFAS if and only if there is a solution to the instance of LEARN that results from the reduction, and therefore we establish that LEARN is NP-hard. In Section 3.4, we

extend our main result to the case when the learning algorithm has access to various oracles, and to the case when there is an upper bound on the number of parents for each node in the solution to LEARN.

For the remainder of this paper we assume—without loss of generality—that in any instance of DBFAS, no vertex has in-degree or out-degree of zero; if such a node exists, none of its incident edges can participate in a cycle, and we can remove that node from the graph without changing the solution.

3.1 A Reduction from DBFAS to LEARN

In this section, we show how to reduce an arbitrary instance of DBFAS into a corresponding instance of LEARN. To help distinguish between elements in the instance of DBFAS and elements in the instance of LEARN, we will subscript the corresponding symbols with D and L, respectively. In particular, we use $\mathcal{G}_D = (\mathbf{V}_D, \mathbf{A}_D)$ and k_D to denote the graph and arc-set bound, respectively, from the instance of DBFAS; from this instance, we create an instance of LEARN consisting of a set of variables $\mathbf{X}_L$, a probability distribution $p_L(\mathbf{X}_L)$, and a parameter bound d_L.

For each $V_i \in \mathbf{V}_D$ in the instance of DBFAS, we create a corresponding nine-state discrete variable V_i for $\mathbf{X}_L$. For each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ in the instance of DBFAS, we create seven discrete variables for $\mathbf{X}_L$: $A_{ij}, B_{ij}, C_{ij}, D_{ij}, E_{ij}, F_{ij}, G_{ij}$. Variables A_{ij}, D_{ij} and G_{ij} have nine states, variables B_{ij}, E_{ij} and F_{ij} have two states, and variable C_{ij} has three states. There are no other variables in $\mathbf{X}_L$ for the instance of LEARN. The probability distribution $p_L(\mathbf{X}_L)$ for the instance of LEARN is specified using a Bayesian network $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$. The model is defined over the variables in $\mathbf{X}_L$, along with, for each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ from the instance of DBFAS, a single "hidden" binary variable H_{ij}. Let $\mathbf{H}_L$ denote the set of all such hidden variables. The distribution $p_L(\mathbf{X}_L)$ is defined by summing the distribution $p_L(\mathbf{H}_L, \mathbf{X}_L)$, defined by $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$, over all of the variables in $\mathbf{H}_L$. The structure $\mathcal{H}_L$ is defined as follows. For each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ in the instance of DBFAS, the DAG contains the edges shown in Figure 1. The number of states for each node in the figure is specified in parentheses below the node.

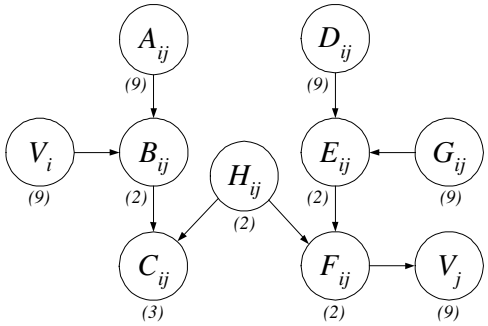
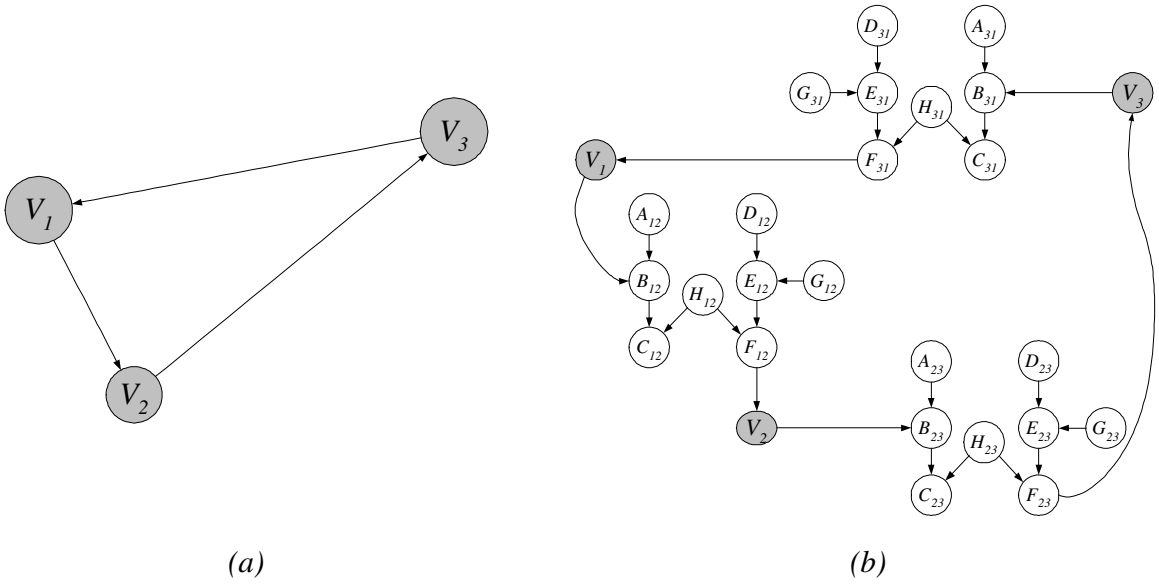


Figure 1: Edges in $\mathcal{H}_L$ corresponding to each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ from the instance of DBFAS. The number of states for each node is given in parentheses below the node.

For an example, Figure 2a shows an instance of DBFAS, and Figure 2b shows the resulting structure of $\mathcal{H}_L$.



(a) *(b)*

Figure 2: An example of the structure $\mathcal{H}_L$ that results from the reduction from a specific instance of DBFAS: (a) an instance of DBFAS consisting of three nodes V_1, V_2 and V_3 and (b) the corresponding structure $\mathcal{H}_L$.

We now specify the local probability distributions in $\theta_{\mathcal{H}_L}$. Let r_X denote the number of states of X, let $\mathbf{Pa}_X$ denote the set of parents of X in $\mathcal{H}_L$, and let $NNZ(\mathbf{pa}_X)$ denote the number of values in $\mathbf{pa}_X$ that are *not* equal to zero. Then for each node X in $\mathcal{H}_L$, the local probability distribution for X is defined as follows:

$$p(X = x|\mathbf{Pa}_X = \mathbf{pa}_X) = \begin{cases} \frac{1}{16} & \text{if } x = 0 \text{ and } NNZ(\mathbf{pa}_X) = 0 \\ \frac{1}{(r_X-1)}\frac{15}{16} & \text{if } x \neq 0 \text{ and } NNZ(\mathbf{pa}_X) = 0 \\ \frac{1}{64} & \text{if } x = 0 \text{ and } NNZ(\mathbf{pa}_X) = 1 \\ \frac{1}{(r_X-1)}\frac{63}{64} & \text{if } x \neq 0 \text{ and } NNZ(\mathbf{pa}_X) = 1 \\ \frac{1}{128} & \text{if } x = 0 \text{ and } NNZ(\mathbf{pa}_X) = 2 \\ \frac{1}{(r_X-1)}\frac{127}{128} & \text{if } x \neq 0 \text{ and } NNZ(\mathbf{pa}_X) = 2. \end{cases} \quad (1)$$

Because each node in $\mathcal{H}_L$ has at most two parents, the above conditions define every local distribution in $\theta_{\mathcal{H}_L}$.

Finally, we define the constant d_L in the instance of LEARN. Every node in $\mathcal{G}_D$ has either exactly one or exactly two parents because, in any instance of DBFAS, the total degree of each node is at most three and by assumption no node has an in-degree or an out-degree of zero. Let t_D denote the number of nodes in $\mathcal{G}_D$ from the instance of DBFAS that have exactly two in-coming edges; similarly, let $o_D = |\mathbf{V}_D| - t_D$ be the number of nodes that have exactly one in-coming edge. Then

we have

$$d_L = 186|\mathbf{A}_D| + 18k_D + 16(|\mathbf{A}_D| - k_D) + 16o_D + 32t_D. \quad (2)$$

We now argue that the reduction is polynomial. It is easy to see that we can specify the structure $\mathcal{H}_L$ and the bound d_L in polynomial time; we now argue that we can specify all of the parameter values $\theta_{\mathcal{H}_L}$ in polynomial time as well. Because each node in $\mathcal{H}_L$ has at most two parents, each corresponding conditional-probability table contains a constant number of parameters. Thus, as long as each parameter is represented using number of bits that is polynomial in the size of the instance of DBFAS, the parameters $\theta_{\mathcal{H}_L}$ can be written down in polynomial time. Each node has either two, three, or nine states, and thus it follows from the specification of $p(X = x|\mathbf{Pa}_X = \mathbf{pa}_X)$ in Equation 1 that each parameter is a fraction whose denominator is a power of two that can never exceed 1024 (i.e., $(9-1) \times 128$). Thus, when using a straight-forward binary representation for the parameter values, we can represent each such value exactly using at most ten (i.e., $\log_2 1024$) bits. Thus we conclude that the entire reduction is polynomial.

3.2 Specifying a Perfect Distribution

In our reduction from DBFAS to LEARN in the previous section, we specified the probability distribution $p_L(\mathbf{X}_L)$ using the Bayesian network $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$. As we shall see in Section 3.3, our proof that LEARN is NP-hard requires that the distribution $p_L(\mathbf{H}_L, \mathbf{X}_L)$ is perfect with respect to the structure $\mathcal{H}_L$. In this section, we discuss the result from the appendices that guarantees that the local distributions defined by Equation 1 lead to an appropriate joint distribution.

Our results on perfectness are closely related to work on qualitative belief networks (QBNs), which are studied by (e.g.) Wellman (1990) and Druzdzel and Henrion (1993). In the appendices, we consider two properties of local probability distributions: one is related to the positive-influence property of QBNs, and the other is related to the positive-product-synergy property of QBNs. For a rigorous definition of these QBN concepts, see Druzdzel and Henrion (1993). Roughly speaking, a distribution has the positive-influence property if observing higher values of a parent node cannot decrease the probability of observing higher values of the target node when all other parent values are fixed. The positive-product-synergy property dictates how changes in the values for a *pair* of parents affects the probability of the target node, and is closely related to the function property *multivariate total positivity of order two* in the mathematics community (see Karlin and Rinott, 1980). The two QBN properties impose *non-strict* inequality constraints. For example, if the local distribution for a node Y has the positive-influence property, then increasing the value of one of its parents does not necessarily increase the probability of Y; it is instead constrained to not decrease. The positive-product-synergy property imposes an analogous non-strict inequality constraint.

In the appendices, we define strict versions of the QBN properties for a special class of distributions. The main result of the appendices (Lemma 17) is that for any Bayesian network in which each local distribution has both of our properties, the joint distribution is necessarily perfect with respect to the network structure. The following result provides a prescription for constructing distributions for which both our properties hold:

Lemma 5 *Let $(\mathcal{G}, \theta_{\mathcal{G}})$ be a Bayesian network, let r_X denote the number of states of node X, let $\mathbf{Pa}_X$ denote the set of parents of node X in $\mathcal{G}$, let NNZ$(\mathbf{pa}_X)$ denote the number of non-zero elements in the set $\mathbf{pa}_X$, and let α_X be a constant satisfying $0 < \alpha_X < 1$. If all of the local distributions in $\theta_{\mathcal{G}}$*

are defined as

$$p(X = x|\mathbf{Pa}_X = \mathbf{pa}_X) = \begin{cases} \alpha_X^{F(\mathbf{pa}_X)} & \text{if } x = 0 \\ \frac{1}{(r_X-1)}\left(1-\alpha_X^{F(\mathbf{pa}_X)}\right) & \text{otherwise,} \end{cases} \quad (3)$$

where

$$F(\mathbf{pa}_X) = 2 - \frac{1}{2}^{NNZ(\mathbf{pa}_X)},$$

then the distribution defined by $(\mathcal{G}, \theta_{\mathcal{G}})$ *is perfect with respect to* $\mathcal{G}$.

The local distributions defined by Equation 1 are simply specializations of Equation 3 where $\alpha_X = \frac{1}{16}$ for every X. Thus, the following corollary follows immediately from Lemma 5:

Corollary 6 *The distribution* $p_L(\mathbf{H}_L, \mathbf{X}_L)$ *resulting from the reduction is perfect with respect to* $\mathcal{H}_L$.

3.3 Reduction Proofs

In this section, we prove LEARN is NP-hard by demonstrating that there is a solution to the instance of DBFAS if and only if there is a solution to the instance of LEARN that results from the reduction. In the results that follow, we often consider sub-graphs of solutions to LEARN that correspond only to those nodes that are "relevant" to a particular arc in the instance of DBFAS. Therefore, to simplify the discussion, we use $\{V_i, V_j\}$ *edge component* to refer to a sub-graph defined by the nodes $\{V_i, A_{ij}, B_{ij}, C_{ij}, D_{ij}, E_{ij}, F_{ij}, G_{ij}, V_j\}$. We use *edge component* without reference to a particular V_i and V_j when an explicit reference is not necessary. Figure 3, which is key to the results that follow, shows two configurations of the edges in an edge component.

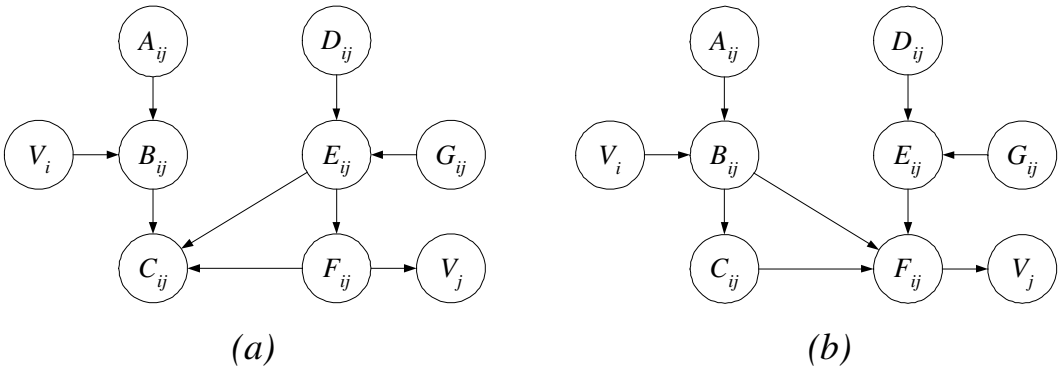


Figure 3: Two configurations of the edges in an edge component.

We first prove a preliminary result that is used in both of the main proofs of this section. Recall that $\mathcal{H}_L$ contains an additional "hidden" node H_{ij} within each edge component. We will be considering active paths in $\mathcal{H}_L$, but are only concerned about those in which the endpoints are in $\mathbf{X}_L$ and for which no H_{ij} is in the conditioning set; these active paths correspond to dependencies that exist within the (marginalized) distribution $p_L(\mathbf{X}_L)$. To simplify presentation, we define a $\mathbf{X}_L$-*restricted* active path to denote such an active path. In this and later results, we will demonstrate

that one DAG $\mathcal{F}_1$ includes another DAG $\mathcal{F}_2$ by showing that for any active path in $\mathcal{F}_2$, there exists a corresponding (i.e., same endpoints and same conditioning set) active path in $\mathcal{F}_1$.

Lemma 7 *Let $p_L(\mathbf{X}_L)$ be the distribution defined for the instance of LEARN in the reduction, and let $\mathcal{F}$ be any DAG defined over $\mathbf{X}_L$ such that each edge component in $\mathcal{F}$ contains the edges in either Figure 3a or in Figure 3b. Then $\mathcal{F}$ includes $p_L(\mathbf{X}_L)$.*

Proof: Let $\mathcal{H}_L$ be the DAG defining $p_L(\mathbf{X}_L)$ in the reduction. We prove that $\mathcal{F}$ includes $p_L(\mathbf{X}_L)$ by demonstrating that for every $\mathbf{X}_L$-restricted active path in $\mathcal{H}_L$, there exists a corresponding active path in $\mathcal{F}$. To do this, we construct an additional model $\mathcal{H}'$ that includes $\mathcal{H}_L$—and consequently $\mathcal{H}'$ can represent $p_L(\mathbf{X}_L)$ exactly—such that $\mathbf{X}_L$-restricted active paths in $\mathcal{H}'$ are easily mapped to their corresponding active paths in $\mathcal{F}$.

We create $\mathcal{H}'$ from $\mathcal{H}_L$ as follows. For each i and j, if the edge component in $\mathcal{F}$ is in the configuration shown in Figure 3a, we add the edge $E_{ij} \rightarrow H_{ij}$ to $\mathcal{H}$ and then reverse the (now covered) edge $H_{ij} \rightarrow F_{ij}$. Similarly, if the edge component in $\mathcal{F}$ is in the configuration shown in Figure 3b, we add the edge $B_{ij} \rightarrow H_{ij}$ to $\mathcal{H}$ and then reverse the edge $H_{ij} \rightarrow C_{ij}$. The resulting components in $\mathcal{H}'$ are shown in Figure 4a and Figure 4b, respectively. Because we created $\mathcal{H}'$ by

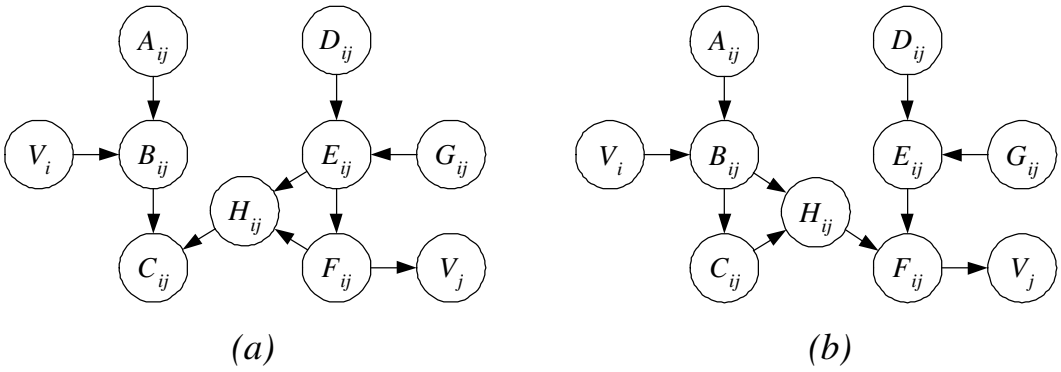


Figure 4: Edges in $\mathcal{H}'$ corresponding to the edge components in Figure 3

edge additions and covered edge reversals, we know by Lemma 4 that $\mathcal{H}'$ includes $\mathcal{H}_L$. It is now easy to see that any $\mathbf{X}_L$-restricted active path in $\mathcal{H}'$ has a corresponding active path in $\mathcal{F}$: simply replace any segment $X \rightarrow H_{ij} \rightarrow Y$ in the path by the corresponding edge $X \rightarrow Y$ from $\mathcal{F}$, and the resulting path will be active in $\mathcal{F}$. □

Theorem 8 *If there is a solution $\mathbf{A}'_D$ to the given instance of DBFAS with $|\mathbf{A}'_D| \leq k_D$, then there is a solution $\mathcal{F}_L$ to the instance of LEARN with $\leq d_L$ parameters.*

Proof: We create a solution DAG $\mathcal{F}_L$ as follows. For every arc $V_i \rightarrow V_j \in \mathbf{A}'_D$ in the DBFAS solution, $\mathcal{F}_L$ contains the edges shown in Figure 3a. For the remaining arcs $V_i \rightarrow V_j$ that are not in $\mathbf{A}'_D$, $\mathcal{F}_L$ contains the edges shown in Figure 3b. $\mathcal{F}_L$ contains no other edges. First we argue that $\mathcal{F}_L$ is acyclic. Each $\{V_i, V_j\}$ edge component in $\mathcal{F}_L$ is itself acyclic, and contains a directed path from V_i to V_j if and only if the corresponding arc $V_i \rightarrow V_j \in \mathbf{A}_D$ from the instance of DBFAS is not in $\mathbf{A}'_D$; if the corresponding arc from the instance of DBFAS is in $\mathbf{A}'_D$, $\mathcal{F}_L$ contains neither

a directed path from V_i to V_j, nor a directed path from V_j to V_i that is contained within the edge component. Therefore, for any hypothetical cycle in $\mathcal{F}_L$, there would be a corresponding cycle in $\mathcal{G}_D$ that passed entirely through arcs not in $\mathbf{A}'_D$, which is impossible assuming $\mathbf{A}'_D$ is a solution to DBFAS. From Lemma 7, we know that $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$. Now we derive the number of parameters supported by $\mathcal{F}_L$. Within each edge component, the parents for A_{ij}, B_{ij}, D_{ij}, E_{ij} and G_{ij} are the same regardless of whether or not the arc is in $\mathbf{A}'_D$; it is easy to verify that for each edge component, the local distributions for these nodes contribute a total of 186 parameters. For each arc $V_i \rightarrow V_j \in \mathbf{A}'_D$, the corresponding nodes C_{ij} and F_{ij} contribute a total of $16+2=18$ parameters; for each arc $V_i \rightarrow V_j \notin \mathbf{A}'_D$, the nodes C_{ij} and F_{ij} contribute a total of $4+12=16$ parameters. For every node $V_i \in \mathbf{V}_D$ in the instance of DBFAS that has exactly two parents, the corresponding $V_i \in \mathbf{X}_L$ in the instance of LEARN will also have two parents. Similarly, for every node $V_i \in \mathbf{V}_D$ with exactly one parent, the corresponding $V_i \in \mathbf{X}_L$ has exactly one parent. By construction of $\mathcal{F}_L$, every parent node for any $V_i \in \mathbf{X}_L$ has two states (and is equal F_{ji} for some j), and therefore because each node $V_i \in \mathbf{X}_L$ has nine states, the total number of parameters used in the local distributions for these nodes is $16o_D + 32t_D$. Thus, we conclude that the number of parameters in $\mathcal{F}$ is exactly

$$186|\mathbf{A}_D| + 18|\mathbf{A}'_D| + 16(|\mathbf{A}_D| - |\mathbf{A}'_D|) + 16o_D + 32t_D.$$

Because $|\mathbf{A}'_D| \leq k_D$, we conclude from Equation 2 that the number of parameters in $\mathcal{F}_L$ is less than or equal to d_L, and thus $\mathcal{F}_L$ is a valid solution to the instance of LEARN. □

Theorem 9 *If there is a solution $\mathcal{F}_L$ to the instance of LEARN with $\leq d_L$ parameters, then there is a solution to the given instance of DBFAS with $|\mathbf{A}'_D| \leq k_D$.*

Proof: Given the solution $\mathcal{F}_L$, we create a new solution $\mathcal{F}_L{}'$ as follows. For every pair (V_i, V_j) corresponding to an edge $V_i \rightarrow V_j \in \mathbf{A}_D$ in the instance of DBFAS, if there is no directed path in $\mathcal{F}_L$ from V_i to V_j, then the corresponding edge component in $\mathcal{F}_L{}'$ contains the edges shown in Figure 3a. Otherwise, when there is at least one directed path in $\mathcal{F}_L$ from V_i to V_j, the corresponding edge component in $\mathcal{F}_L{}'$ contains the edges shown in Figure 3b. By construction, $\mathcal{F}_L{}'$ will contain a cycle only if $\mathcal{F}_L$ contains a cycle, and consequently we conclude that $\mathcal{F}_L{}'$ is a DAG. From Lemma 7, we know that $\mathcal{F}_L{}'$ includes $p_L(\mathbf{X}_L)$.

In the next two paragraphs, we argue that $\mathcal{F}_L{}'$ does not support more parameters than does $\mathcal{F}_L$. Consider the DAG $\mathcal{F}^0$ that is identical to $\mathcal{F}_L{}'$, except that for all i and j, the only parent of C_{ij} is B_{ij} and the only parent of F_{ij} is E_{ij} (see Figure 5). Because $\mathcal{F}^0$ is a subgraph of $\mathcal{F}_L{}'$, any active

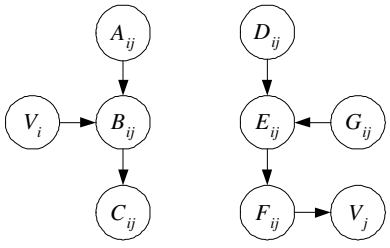


Figure 5: Edges within each edge component of $\mathcal{F}^0$

path in $\mathcal{F}^0$ must have a corresponding active path in $\mathcal{F}_L{}'$, and thus we conclude that $\mathcal{F}_L{}'$ includes

$\mathcal{F}^0$. The original solution $\mathcal{F}_L$ also includes $\mathcal{F}^0$ by the following argument: $\mathcal{F}^0$ is a strict sub-graph of $\mathcal{H}_L$ ($\mathcal{F}^0$ contains a subset of the edges and no H_{ij} nodes), and thus any active path in $\mathcal{F}^0$ has a corresponding $\mathbf{X}_L$-restricted active path in $\mathcal{H}_L$; because $\mathcal{H}_L$ is perfect with respect to the distribution $p_L(\mathbf{H}_L, \mathbf{X}_L)$ defined by $(\mathcal{H}_L, \theta_{\mathcal{H}_L})$ (Corollary 6), we know that any such $\mathbf{X}_L$-restricted active path in $\mathcal{H}_L$ corresponds to a dependence in $p_L(\mathbf{X}_L)$, and thus, because $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$, there must be a corresponding active path in $\mathcal{F}_L$.

From Theorem 3, we know that there exists a sequence of edge additions and covered edge reversals that transforms $\mathcal{F}^0$ into $\mathcal{F}_L$, and another sequence of edge additions and covered edge reversals that transforms $\mathcal{F}^0$ into $\mathcal{F}_L{}'$. From Lemma 1 and Lemma 2, a covered edge reversal does not change the number of parameters supported by a DAG, and thus we can compare the number of parameters supported by the two DAGs by evaluating the increase in parameters that result from the additions within each of the two transformations. $\mathcal{F}^0$ can be transformed into $\mathcal{F}_L{}'$ by simply adding, for each edge component, the corresponding two extra edges in $\mathcal{F}_L{}'$. That is, we either (1) add the edges $E_{ij} \rightarrow C_{ij}$ and $F_{ij} \rightarrow C_{ij}$, resulting in an increase of 12 parameters, or (2) add the edges $B_{ij} \rightarrow F_{ij}$ and $C_{ij} \rightarrow F_{ij}$, resulting in an increase of 10 parameters. If $\mathcal{F}_L$ supports fewer parameters than $\mathcal{F}_L{}'$, there must be at least one $\{V_i, V_j\}$ edge component for which the total parameter increase from adding edges between nodes in that component is less than the corresponding increase in $\mathcal{F}_L{}'$. In order to reverse any edge in an edge component from $\mathcal{F}^0$, we need to first cover that edge by adding at least one other edge that is contained in that component; it is easy to verify that any such "covering addition" results in an increase of at least 16 parameters (adding $E_{ij} \rightarrow V_j$ results in this increase, and all other additions result in a larger increase). Thus we conclude that for the $\{V_i, V_j\}$ edge component, only edge additions are performed in the transformation from $\mathcal{F}^0$ to $\mathcal{F}_L$. H_{ij} does not exist in $\mathcal{F}_L$, and therefore because $p_L(\mathbf{H}_L, \mathbf{X}_L)$ is a DAG-perfect distribution (Corollary 6), C_{ij} and F_{ij} cannot be conditionally independent given any other nodes in $\mathbf{X}_L$; thus, in order for $\mathcal{F}_L$ to include $p_L(\mathbf{X}_L)$, there must be an edge between C_{ij} and F_{ij} in $\mathcal{F}_L$. We consider two cases, corresponding to the two possible directions of the edge between C_{ij} and F_{ij} in $\mathcal{F}_L$. If the edge is directed as $C_{ij} \rightarrow F_{ij}$, we know that there is a directed path between V_i and V_j in $\mathcal{F}_L$ because none of the edges from $\mathcal{F}^0$ can be reversed. By construction of $\mathcal{F}_L{}'$, this implies that the increase in parameters supported by $\mathcal{F}_L{}'$ attributed to this edge component is 10. In $\mathcal{H}_L$, F_{ij} and B_{ij} are d-connected given any conditioning set from $\mathbf{X}_L$ that contains C_{ij} (see Figure 1), and thus we know that $F_{ij} \not\perp\!\!\!\perp_{p_L} B_{ij} | \mathbf{S}$ for any $\mathbf{S} \subset \mathbf{X}_L$ that contains C_{ij}; this implies that the edge $B_{ij} \rightarrow F_{ij}$ must exist in $\mathcal{F}_L$, else we could find a conditioning set $\mathbf{S}$ that contains C_{ij} for which $F_{ij} \perp\!\!\!\perp_{\mathcal{F}_L} B_{ij} | \mathbf{S}$, which contradicts the fact that $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$. But adding both $C_{ij} \rightarrow F_{ij}$ and $B_{ij} \rightarrow F_{ij}$ to $\mathcal{F}^0$ requires an addition of *at least* 10 parameters, contradicting the supposition that the parameter increase due to this edge component is smaller in $\mathcal{F}_L$ than in $\mathcal{F}_L{}'$. If the edge between C_{ij} and F_{ij} is directed as $F_{ij} \rightarrow C_{ij}$, we know that $\mathcal{F}_L$ must also contain the edge $E_{ij} \rightarrow C_{ij}$, lest (using the same logic as above) there would exist some conditioning set $\mathbf{S}$ containing F_{ij} such that $C_{ij} \perp\!\!\!\perp_{\mathcal{F}_L} E_{ij} | \mathbf{S}$ but $C_{ij} \not\perp\!\!\!\perp_{p_L} E_{ij} | \mathbf{S}$, contradicting the fact that $\mathcal{F}_L$ includes $p_L(\mathbf{X}_L)$. Adding both of these edges, however, requires an addition of *at least* 12 parameters; because the corresponding edge component in $\mathcal{F}_L{}'$ attributed *at most* 12 parameters in the transformation from $\mathcal{F}_L{}'$, this again contradicts the supposition that the parameter increase due to this edge component is smaller in $\mathcal{F}_L$ than in $\mathcal{F}_L{}'$.

Having established that $\mathcal{F}_L{}'$ is a solution to LEARN that supports fewer parameters than $\mathcal{F}_L$, we now use $\mathcal{F}_L{}'$ to construct a solution $\mathbf{A}'_D$ to the instance of DBFAS. For each $\{V_i, V_j\}$ edge

component in ${\mathcal{F}_L}'$, if that component contains the edges shown in Figure 3a, then we include in $\mathbf{A}'_D$ the arc $V_i \rightarrow V_j$. $\mathbf{A}'_D$ contains no other arcs.

We now argue that $\mathbf{A}'_D$ contains at least one arc from every cycle from the instance of DBFAS. Each arc $V_i \rightarrow V_j \in \mathbf{A}_D$ that is *not* contained in $\mathbf{A}'_D$ has a corresponding edge component in ${\mathcal{F}_L}'$ for which there is a directed path from V_i to V_j. Thus, any hypothetical cycle in the instance of DBFAS that does not pass through an edge in $\mathbf{A}'_D$ has a corresponding directed cycle in ${\mathcal{F}_L}'$, which is impossible because ${\mathcal{F}_L}'$ is a DAG.

Finally, we argue that $\mathbf{A}'_D$ contains at most k_D arcs. Recall that o_D and t_D denote the number of nodes in $\mathcal{G}_D$ that have exactly one and two in-coming edges, respectively. As in the proof of Theorem 8, it is easy to verify that the number of parameters d'_L supported by ${\mathcal{F}_L}'$ is exactly

$$186|\mathbf{A}_D| + 18|\mathbf{A}'_D| + 16(|\mathbf{A}_D| - |\mathbf{A}'_D|) + 16o_D + 32t_D.$$

Given that $d'_L \leq d_L$, we conclude from Equation 2 that $|\mathbf{A}'_D| \leq k_D$. □

Given the previous results, the main result of this paper now follows easily.

Theorem 10 *LEARN is NP-hard.*

Proof: Follows immediately from Theorem 8 and Theorem 9. □

Also, due to the fact that the distribution in the reduction is obtained by marginalizing out the hidden variables in a DAG-perfect distribution, the following result is immediate.

Corollary 11 *LEARN remains NP-hard when we restrict the input probability distribution to be the marginalization of a DAG-perfect distribution.*

3.4 Extensions

Many approaches to learning Bayesian networks from data use independence tests or mutual-information calculations to help guide a search algorithm. In this section, we show that even if such tests and calculations could be obtained in constant time, the search problem remains hard. In particular, we show that Theorem 10 holds even when the learning algorithm has access to at least one of three oracles. Furthermore, we show that the problem remains hard when we restrict ourselves to considering only those solutions to LEARN for which each node has at most k parents, for all $k \geq 3$.

The first oracle we consider is an independence oracle. This oracle can evaluate independence queries in constant time.

Definition 12 (Independence Oracle)
An independence oracle *for a distribution $p(\mathbf{X})$ is an oracle that, in constant time, can determine whether or not $X \perp\!\!\!\perp_p Y|\mathbf{Z}$ for any X and Y in $\mathbf{X}$ and for any $\mathbf{Z} \subseteq \mathbf{X}$.*

The second oracle we consider can perform certain inference queries in constant time; namely, the inference oracle can return the joint probability of any constant-sized set of variables. This oracle can in turn be used to compute conditional probabilities in constant time using division.

Definition 13 (Constrained Inference Oracle)
A constrained inference oracle *for a distribution $p(\mathbf{X})$ is an oracle that, in constant time, can compute $p(\mathbf{Z} = \mathbf{z})$ for any $\mathbf{Z} \subseteq \mathbf{X}$ such that $|\mathbf{Z}| \leq k$ for some constant k.*

Some learning algorithms use mutual information—or an approximation of mutual information—from a distribution to help construct model structures. The *(conditional mutual) information* between variables X and Y given the set of variables $\mathbf{Z}$ is defined as

$$Inf(X;Y|\mathbf{Z}) = \sum_{x,y,\mathbf{z}} p(x,y,\mathbf{z}) \log \frac{p(x,y|\mathbf{z})}{p(x|\mathbf{z})p(y|\mathbf{z})}. \tag{4}$$

The third oracle we consider can compute the mutual information between two variable in constant time, given that there are only a constant number of variables in the conditioning set.

Definition 14 (Constrained Information Oracle)
A constrained information oracle *for a distribution $p(\mathbf{X})$ is an oracle that, in constant time, can compute $Inf(X;Y|\mathbf{Z})$ for any X and Y in $\mathbf{X}$ and for any $\mathbf{Z} \subseteq \mathbf{X}$ such that $|\mathbf{Z}| \leq k$ for some constant k.*

Theorem 15 *Theorem 10 holds even when the learning algorithm has access to (1) an independence oracle, (2) a constrained inference oracle, or (3) a constrained information oracle.*

Proof: We establish this result by demonstrating that we can implement all three of these oracles in polynomial time using the Bayesian network $(\mathcal{H}, \theta_{\mathcal{H}})$ from our reduction. Thus if LEARN can be solved in polynomial time when we have access to any of the constant-time oracles, it must also be solvable in polynomial time *without* any such oracle.

(1) holds immediately because we can test for d-separation in $\mathcal{H}$ in polynomial time. (3) follows from (2) because, given that each variable has some constant number of states, we can implement a constrained information oracle via Equation 4 by calling a constrained inference oracle a constant number of times.

Let $\mathbf{Z} \subseteq \mathbf{X}$ be any subset of the variables such that $|\mathbf{Z}| \leq k$ for some constant k. It remains to be shown how to compute $p(\mathbf{Z} = \mathbf{z})$ in polynomial time from $(\mathcal{H}, \theta_{\mathcal{H}})$. The trick is to see that there is always a cut-set of constant size that decomposes $\mathcal{H}$ into a set of polytrees, where each polytree has a constant number of nodes; within any polytree containing a constant number of nodes, we can perform inference in constant time. We define a cut-set $\mathbf{B}$ as follows: $\mathbf{B}$ contains every node B_{ij} for which (1) C_{ij} is in $\mathbf{Z}$ and (2) B_{ij} is not in $\mathbf{Z}$. Note that $\mathbf{B} \cap \mathbf{Z} = \emptyset$. Given conditioning set $\mathbf{B}$, no active path can contain a node C_{ij} as an interior (i.e., non-endpoint) node, even when any subset of $\mathbf{Z}$ is added to the conditioning set (see Figure 6): any such hypothetical active path must pass through at least one segment $B_{ij} \rightarrow C_{ij} \leftarrow H_{ij}$. But this is not possible, because every such segment is blocked: if C_{ij} is not in $\mathbf{Z}$, then the segment is blocked because C_{ij} has no descendants, and hence can have no descendants in the conditioning set; if C_{ij} is in $\mathbf{Z}$, then we know that $B_{ij} \in \mathbf{B}$ and thus the segment is blocked by B_{ij}.

Because no active path can pass through a node C_{ij}, it follows by construction of $\mathcal{H}$ that—given $\mathbf{B}$ and any subset of $\mathbf{Z}$—each node in $\mathbf{Z}$ is d-connected to only a constant number of other nodes in $\mathbf{Z}$. Furthermore, the structure of $\mathcal{H}$ that is bounded between the C_{ij} nodes forms a polytree. Thus, we can express $p(\mathbf{Z} = \mathbf{z})$ as

$$\begin{aligned} p(\mathbf{Z} = \mathbf{z}) &= \sum_{\mathbf{b}} p(\mathbf{Z} = \mathbf{z}, \mathbf{B} = \mathbf{b}) \\ &= \sum_{\mathbf{b}} \prod_{i} p(\mathbf{T}_i = \mathbf{t}_i(\mathbf{z}, \mathbf{b})), \end{aligned}$$

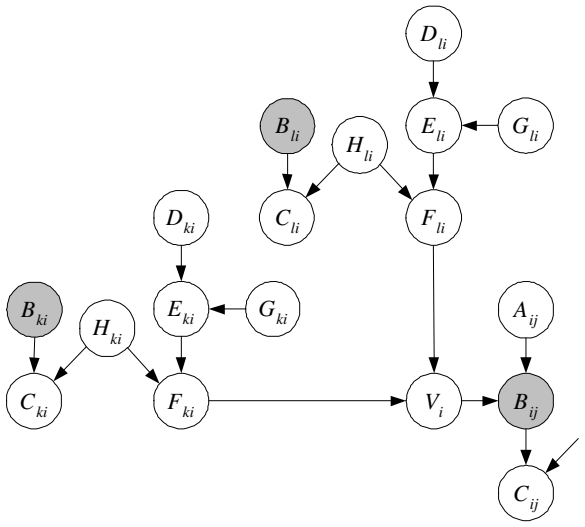


Figure 6: Portion of $\mathcal{H}$ showing that no active path can pass through any C_{ij} once B_{ij} is given.

where each $\mathbf{T}_i$ contains a constant number of variables—$\mathbf{t}_i(\mathbf{z},\mathbf{b})$ is the set of values for those variables as determined by $\mathbf{z}$ and $\mathbf{b}$—that constitute a polytree in $\mathcal{H}$. Thus, each term $p(\mathbf{T}_i = \mathbf{t}_i(\mathbf{z},\mathbf{b}))$ above can be computed in constant time using inference in a polytree. Because there are at most k nodes in $\mathbf{Z}$, the set $\mathbf{B}$ can contain at most k nodes. Therefore, given that each node in $\mathbf{B}$ has at most r states, there are at most r^k terms in the sum above—where both r and k are constants—and we conclude that $p(\mathbf{Z})$ can be computed in polynomial time. □

Finally, we prove that if we restrict LEARN to solutions in which each node has at most k parents, the problem remains NP-hard for all $k \geq 3$.

Theorem 16 *Theorem 15 holds even when solutions to LEARN are restricted to DAGs in which each node has at most k parents, for all* $k \geq 3$.

Proof: The case where $k = 3$ follows immediately from the proof of Theorem 8, where the constructed solution to LEARN is a DAG in which each node has at most three parents, and from the proof of Theorem 9, where the given solution to LEARN is converted into a (better) solution in which each node has at most three parents. It is easy to see that these proofs remain valid under a less restrictive ($k > 3$) bound on the number of parents, and thus the theorem follows. □

4. Conclusion

In this paper, we demonstrated that the problem of identifying high-scoring DAGs from large datasets when using a consistent scoring criterion is NP-hard. Together with the result of Chickering (1996) that the non-asymptotic learning problem is NP-hard, our result implies that learning is hard regardless of the size of the data. There is an interesting gap in the present results. In particular, Chickering (1996) proved that finite-sample learning is NP-hard when each node is restricted to have at most two parents, whereas in this paper we proved that large-sample learning is NP-hard with a three-parent restriction. This leads to the question of whether or not large-sample learning is

NP-hard when we restrict to two parents; we believe that this problem is probably NP-hard, and is worth further investigation.

In practice, the large-sample learning problem actually requires scanning a dataset with a large number of samples, as opposed to accessing a compact representation of the generative distribution. We could alternatively have defined a learning problem in which there is an actual data set supplied; the problem with this approach is that in order to guarantee that we get the large-sample ranking of model structures, we will need the number of data points to be so large that the size of the problem instance is exponential in the number of variables in the domain. Our results have practical importance when it is reasonable to assume that (1) there is enough data such that the relative ranking of those DAGs considered by the learning algorithm is the same as in the large-sample limit, and (2) the number of records in the data is small enough that we can compute the score for candidate structures in a reasonable amount of time.

As discussed in Section 1, there exist assumptions about the generative distribution that lead to efficient large-sample learning algorithms. These assumptions are not likely to hold in most real-world scenarios, but the corresponding "correct" algorithms can work well even if the assumptions do not hold. An interesting line of research is to investigate alternative, weaker assumptions about the generative distribution that lead to efficient learning algorithms and guarantee large-sample correctness.

Appendix A. Introduction to Perfectness Proofs

As described in Section 3.2, in these appendices we define two properties of local distributions, and prove that as long as these properties hold for every local distribution in the network, the corresponding joint distribution is perfect with respect to the network structure. Lemma 5 follows immediately from our main result once we demonstrate that the two properties hold for the family of distributions defined by Equation 3.

The two properties that we define are *binary-like lattice* (BLL) and *binary-like totally strictly positive* (BLTSP). The "binary like" aspect of both of these properties refers to the fact that the distributions are defined such that we can treat each variable *as if* it only has two states: a "distinguished" state and an "other" state. As first mentioned in Section 3.2, the BLL property of a local distribution is similar to the positive-influence property found in the QBN literature; it specifies that the probability of a node being in its "distinguished" state necessarily increases when we change a single parent from the "other" state to the "distinguished" state. The difference between BLL and the positive-influence property is that BLL requires that the probability strictly increase, whereas the positive-influence property requires that the probability does not decrease. The BLTSP property of a local distribution is similar to the positive-synergy property in the QBN literature. The intuition behind this property is that it requires that the (BLL) influence of a parent strictly increases with the number of other parents that are in the "distinguished" state. The difference between BLTSP and positive synergy is, as above, the requirement of a strict inequality.

Our main result demonstrates that if all local distributions in a Bayesian network are both BLL and BLTSP, then any active path corresponds to a dependence in the joint probability distribution defined by that network:

Lemma 17 *Let $(\mathcal{G}, \theta)$ be a Bayesian network in which all local distributions defined by θ are both BLL and BLTSP. Then the joint distribution represented by $(\mathcal{G}, \theta)$ is perfect with respect to $\mathcal{G}$.*

The proof of Lemma 17 is non-trivial, but the main technique can be understood as follows. We prove perfectness by demonstrating that for any active path between two nodes X and Y, there is a corresponding dependence in the joint distribution whenever the nodes in the conditioning set are all in their distinguished states. If X and Y are adjacent in $\mathcal{G}$ and if there are no nodes in the conditioning set, this dependence follows easily from the definition of BLL. We establish the general result by induction, using the simple case as the basis. In general, we show how to apply graph transformations that result in a simpler model for which our induction step applies. Each graph transformation is defined such that the original distribution over the non-observed nodes, when conditioned on the observed nodes, is represented exactly, and where every local distribution retains the BLL property; the BLTSP property is required in the original distributions to guarantee that the BLL property is retained as a result of each transformation.

The appendices are organized as follows. In Appendix B, we describe graph-transformation methods that can be applied to a Bayesian network. In Appendix C, we rigorously define BLL and BLTSP, and we demonstrate that the transformations described in Appendix B necessarily maintain the BLL property on every distribution. Finally, in Appendix D, we prove Lemma 17.

To simplify notation, we use $\mathcal{G}$ to denote a Bayesian network (as opposed to just the structure of that network) for the remainder of the paper, and we leave the parameter values θ implicit.

Appendix B. Graph Transformations

In this section, we describe a number of transformations that we apply to a Bayesian network in order to more easily prove our main result. For the remainder of the paper, we will assume that the domain of interest $\mathbf{V} = \{V_1, \ldots, V_n\}$ is decomposed into two sets of variables: $\mathbf{O}$ is the set of observed variables for which we are given a corresponding set of states $\mathbf{o}$, and $\mathbf{U}$ is the set of unobserved variables. In contrast to the "hidden" variables $\mathbf{H}_L$ described in Section 3.1, the unobserved variables $\mathbf{U}$ simply correspond to variables that are not in the particular conditioning set $\mathbf{O}$.

Given a Bayesian network $\mathcal{G}$ defined over the domain $\mathbf{O} \cup \mathbf{U}$, each transformation outputs a new Bayesian network $\mathcal{G}^T$. We will use $p(\cdot)$ and $p^T(\cdot)$ to denote the probability distributions defined by $\mathcal{G}$ and $\mathcal{G}^T$, respectively. As we see below, $\mathcal{G}^T$ may be defined over only a subset of the nodes in $\mathcal{G}$. Using $\mathbf{O}^T$ and $\mathbf{U}^T$ to denote the observed and unobserved nodes, respectively, that remain in $\mathcal{G}^T$, all of our transformations maintain the following invariant:

$$\forall \mathbf{u}^T \qquad p^T(\mathbf{u}^T | \mathbf{o}^T) = p(\mathbf{u}^T | \mathbf{o}). \tag{5}$$

Note that $\mathbf{o}$ and $\mathbf{o}^T$ are fixed (observed) values. In words, Equation 5 asserts that the distribution over the unobserved nodes that remain after the transformation is identical in the two models whenever we condition on the observed values.

B.1 Individual Transformations

There are five transformations that we use to prove our main result: edge deletion, edge reversal, node combination, barren node removal, and observed-child separation. For each transformation, there is both a structural change (e.g., an edge $X \rightarrow Y$ is added) and a corresponding change to the conditional distributions (e.g., the local distribution for Y is extended to include the new parent X). For the remainder of this paper, we assume that the local distributions in the model that result

from a transformation are obtained via inference from the original model. In the case where the parents of a node are identical in the two models, inference corresponds to copying the original local distribution.

Whenever the structure of the resulting model includes the original distribution, populating the local distributions via inference results in a new model that defines the original joint distribution. This follows because, by definition of inclusion, there exists a set of local distributions for the new model that yield the original joint distribution. Furthermore, these local distributions are unique (assuming that the original distribution is positive) and must match the corresponding conditional distributions from the original model; we use inference to ensure this match.

We say that a transformation is *valid* if (1) the preconditions of the transformation (e.g., there exists an edge $X \rightarrow Y$ in the model) are met and (2) the result of the transformation is an acyclic model. We now consider each transformation in turn.

B.1.1 EDGE DELETION

An *edge deletion* deletes an edge of the form $O \rightarrow Y$ from $\mathcal{G}$, where $O \in \mathbf{O}$ is an observed node, and replaces the local distribution in Y by the same local distribution except that the value o for O is fixed to its observed value (e.g.) o^0 (see Figure 7). Thus, if the parents of Y are $O \cup \mathbf{Z}$ in $\mathcal{G}$, then the new local distribution for Y in $\mathcal{G}^T$ is defined as

$$p^T(y|\mathbf{z}) = p(y|\mathbf{z}, o^0),$$

where the probability $p(y|\mathbf{z}, o^0)$ can be extracted directly from the local distribution of Y in $\mathcal{G}$. It is easy to see that for the resulting model $\mathcal{G}^T$ we have

$$\forall \mathbf{u} \qquad p^T(\mathbf{u}, \mathbf{o}) = p(\mathbf{u}, \mathbf{o})$$

(for fixed $\mathbf{o}$) and thus Equation 5 holds. Because deleting an edge can never create a cycle, an edge-deletion transformation (for an existing edge) is always valid.

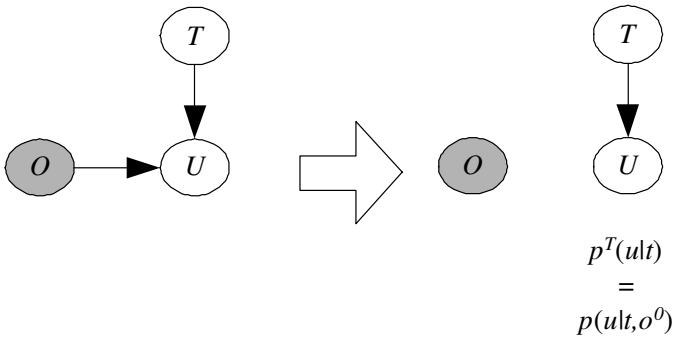


Figure 7: Example of an edge-deletion transformation. The observed value of node O is o^0. After deleting the edge $O \rightarrow U$, the local distribution for node U is identical to the original distribution when constrained to $o = o^0$.

B.1.2 EDGE REVERSAL

An *edge reversal*, originally defined by Howard and Matheson (1981), is a transformation that first "covers" an edge by adding new edges until the edge is covered, and then reverses the edge (see Figure 8). In particular, for an edge $H \rightarrow Y$, let $\mathbf{X}$ be the parents of H that are not parents of Y, let $\mathbf{Z}$ be the parents of both H and Y, and let $\mathbf{W}$ be the parents of Y that are not parents of H. The edge-reversal transformation adds the edge $X \rightarrow Y$ for every $X \in \mathbf{X}$, adds the edge $W \rightarrow H$ for every $W \in \mathbf{W}$, and reverses $H \rightarrow Y$.

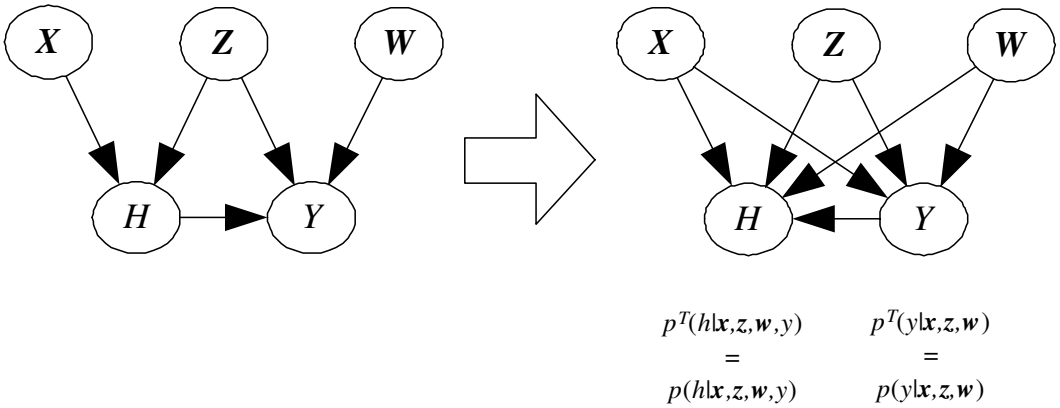


Figure 8: The relevant fragment of a graph structure for an edge-reversal transformation.

As shown in Figure 8, the local distributions for H and Y in $\mathcal{G}^T$ are defined by the joint distribution defined in $\mathcal{G}$. In contrast to the edge-deletion transformation, the local probability distributions for these nodes in $\mathcal{G}^T$ cannot simply be extracted from the corresponding distributions in $\mathcal{G}$; for example, we obtain the local distribution for H in $\mathcal{G}^T$ via inference as follows:

$$\begin{aligned} p^T(h|\mathbf{x},\mathbf{z},\mathbf{w},y) &= p(h|\mathbf{x},\mathbf{z},\mathbf{w},y) \\ &= \frac{p(h,y|\mathbf{x},\mathbf{z},\mathbf{w})}{p(y|h,\mathbf{x},\mathbf{z},\mathbf{w})} \\ &= \frac{p(h|\mathbf{x},\mathbf{z})p(y|h,\mathbf{z},\mathbf{w})}{\sum_i p(h^i|\mathbf{x},\mathbf{z})p(y|h^i,\mathbf{z},\mathbf{w})}, \end{aligned}$$

where $p(h|\mathbf{x},\mathbf{z})$ and $p(y|h,\mathbf{z},\mathbf{w})$ are the local distributions in $\mathcal{G}$.

Proposition 18 *If there is no directed path from H to Y other than the edge $H \rightarrow Y$, then the edge can be covered as described without creating a cycle.*

Proof: Suppose not. Using the notation from above, there must either be some $X \in \mathbf{X}$ for which adding $X \rightarrow Y$ creates a cycle, or there must be some $W \in \mathbf{W}$ for which adding $W \rightarrow H$ creates a cycle. Because there is already a directed path from X to Y, we immediately rule out the first case. If adding $W \rightarrow H$ creates a cycle, then there must already be a directed path from H to W. By appending $W \rightarrow Y$ to this directed path, we have a directed path from H to Y that is not the edge $H \rightarrow Y$, yielding a contradiction. □

Because no independence constraints are added as a result of adding an edge to a model, and because the edge is reversed only after being covered, the following result follows immediately from Proposition 18 and Lemma 1:

Proposition 19 *If in $\mathcal{G}$ there is no directed path from H to Y other than the edge $H \rightarrow Y$, then the edge-reversal transformation applied to $H \rightarrow Y$ is valid; and for the model $\mathcal{G}^T$ that results, the constraints of Equation 5 must hold.*

B.1.3 NODE COMBINATION

A *node combination* takes a set of nodes $\mathbf{Y}$, where each node in $\mathbf{Y}$ has no children, and replaces the set with the single *composite* node $Y = Comp(\mathbf{Y})$ whose states take on the cross product of the states of all nodes in $\mathbf{Y}$ (see Figure 9). The parents of Y are defined to be the union of all of the parents of the nodes in $\mathbf{Y}$. Because no node in $\mathbf{Y}$ has any children, it is easy to see that applying a node-combination transformation can never create a cycle, and thus the transformation is always valid.

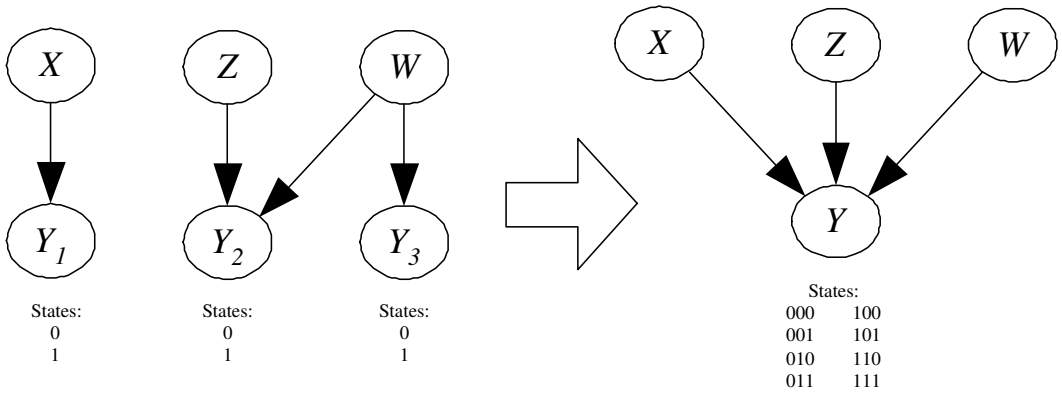


Figure 9: An example of a node-combination transformation. The state space of the combined node Y has a unique state for every possible combination of values for Y_1, Y_2, and Y_3.

The local distribution for the composite node Y is defined in the obvious way: the local probability in $\mathcal{G}^T$ of a composite state y given the parent values is simply the joint probability of the corresponding states of $\mathbf{Y}$ in the original model $\mathcal{G}$ given the same parent values.

Although the set of nodes in the Bayesian network $\mathcal{G}^T$ that results from a node combination is different than in the original network $\mathcal{G}$, it is important to understand that $\mathcal{G}^T$ represents a probability distribution over the original set of nodes. In particular, because the states of the composite node Y are defined to be the cross product of the states for all of the nodes in $\mathbf{Y}$, there is a one-to-one correspondence between states of Y and sets of all states of the nodes in $\mathbf{Y}$. Thus, given any Bayesian network containing composite nodes, we can always "unwind" those nodes into a clique of nodes, where each node in the clique—in addition to the adjacencies within the clique—has the same parents and children of the composite node. Because the nodes that made up the composite node form a clique, there are no independence constraints introduced by this unwinding process. For the remainder of this paper, when we discuss the joint distribution represented by a Bayesian

network, it is to be understood that we mean the distribution over the original domain; we leave implicit the unwinding process that can be performed so that the networks contain the same nodes.

B.1.4 BARREN NODE REMOVAL

An unobserved node $U \in \mathbf{U}$ is *barren* if U has no children. An observed node $O \in \mathbf{O}$ is barren if O has no parents and no children. The *barren-node-removal* transformation simply removes from $\mathcal{G}$ any barren nodes along with their incident edges (see Figure 10). Because a barren node has no children, no conditional distributions change (other than the deletion of the barren-node distribution). Because removing a barren node can never create a cycle, the transformation is always valid.

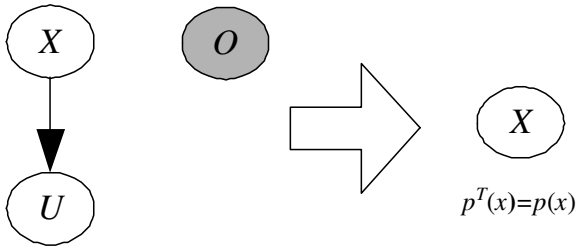


Figure 10: An example of a barren-node-removal transformation: both the unobserved node U and the observed node O are barren.

We now explain why Equation 5 must hold after removing an *unobserved* barren node. Letting $\mathbf{U}^T = \mathbf{U} \setminus U$ denote the unobserved nodes that remain after removing U, we can compute the joint probability in the original model over $\mathbf{U}^T$ and $\mathbf{O}$ as

$$p(\mathbf{u}^T, \mathbf{o}) = \sum_u p(u, \mathbf{u}, \mathbf{o}).$$

Because U has no children, we can "push the sum" all the way through to the last conditional distribution:

$$\begin{aligned} p(\mathbf{u}^T, \mathbf{o}) &= \prod_{x \in \mathbf{u}^T \cup \mathbf{o}} p(x|\mathbf{pa}^X) \left(\sum_u p(u|\cdot) \right) \\ &= \prod_{x \in \mathbf{u}^T \cup \mathbf{o}} p(x|\mathbf{pa}^X). \end{aligned}$$

Because $\mathcal{G}^T$ is identical to $\mathcal{G}$ except that it does not contain node U, it follows that the above product of conditional distributions is exactly the distribution represented by $\mathcal{G}^T$; thus $p^T(\mathbf{u}^T, \mathbf{o}) = p(\mathbf{u}^T, \mathbf{o})$ and Equation 5 must hold.

Equation 5 holds after removing an *observed* barren node O by a similar argument and because O is independent of every other node regardless of the conditioning set.

B.1.5 OBSERVED CHILD SEPARATION (OCS)

The observed-child-separation (OCS) transformation is a "macro" transformation that combines a node-combination transformation, an edge-reversal transformation, and an edge-deletion transfor-

mation. In particular, let H be any node in $\mathbf{U}$ that has at least one child in $\mathbf{O}$, and let $\mathbf{Y} = \{Y_1, \ldots, Y_m\}$ denote the set of all children of H that are in $\mathbf{O}$ that have no children themselves. For the OCS transformation (see the example in Figure 11), we first apply a node-combination transformation to the nodes in $\mathbf{Y}$, resulting in a model containing the composite node $Y = Comp(\mathbf{Y})$. Next, we apply an edge-reversal transformation on the edge $H \rightarrow Y$. Finally, we delete the resulting edge $Y \rightarrow H$ using an edge-deletion transformation. The OCS transformation is valid whenever the sub-transformations are valid.

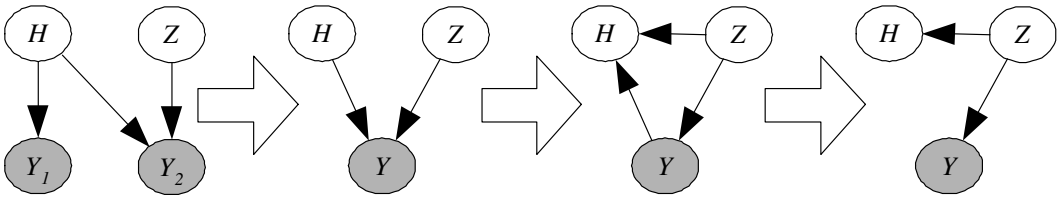


Figure 11: An example showing the sub-transformations that make up the OCS macro transformation.

Because we have already shown that the invariant of Equation 5 holds after each component transformation that makes up the OCS macro transformation, we conclude that Equation 5 holds as a result of this transformation as well.

In Figure 12, we show the relevant network fragment both before and after a general OCS transformation is applied, along with the local distributions in $\mathcal{G}^T$ that must be derived via inference in $\mathcal{G}$. The nodes Y_j—which are shaded in the figure—are the observed children of node H that will be separated from H using the transformation. The bold-font nodes $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{W}$ represent sets of nodes: each node in $\mathbf{X}$ is a parent of H but not a parent of any Y_j, each node in $\mathbf{Z}$ is a parent of H and a parent of at least one Y_j, and each node in $\mathbf{W}$ is not a parent of H but is a parent of at least one Y_j. In the figure, the term $\mathbf{y^0}$ in $p(h|\mathbf{x}, \mathbf{z}, \mathbf{w}, \mathbf{y^0})$ is shorthand for the set of all observed states $y_1^0, \ldots, y_n^0$ of the Y_j variables; we assume that y_j^0 is the observed state of Y_j for all j. Similarly, the term $\mathbf{y}$ in $p(\mathbf{y}|\mathbf{x}, \mathbf{z}, \mathbf{w})$ denotes an arbitrary set of states $y_1, \ldots, y_n$ for the observed Y_j variables.

B.2 Transformation Algorithms

In this section, we present two graph-transformation algorithms that, like the OCS "macro" transformation, apply a sequence of transformations to a model $\mathcal{G}$. We distinguish an "algorithm" from a "macro transformation" by the fact that in the former, the order in which we apply the individual transformations depends on the topology of the entire network structure. As in the case of the OCS macro transformation, we conclude that because the individual transformations that define the algorithm all maintain the invariant of Equation 5, the invariant holds for the algorithms as well.

We say that a node X in a graph is a *lowest* node with some property if no descendant of X in the graph also has that property. Thus when the graph is a DAG containing at least one node with a given property, there must always exist at least one lowest node with that property.

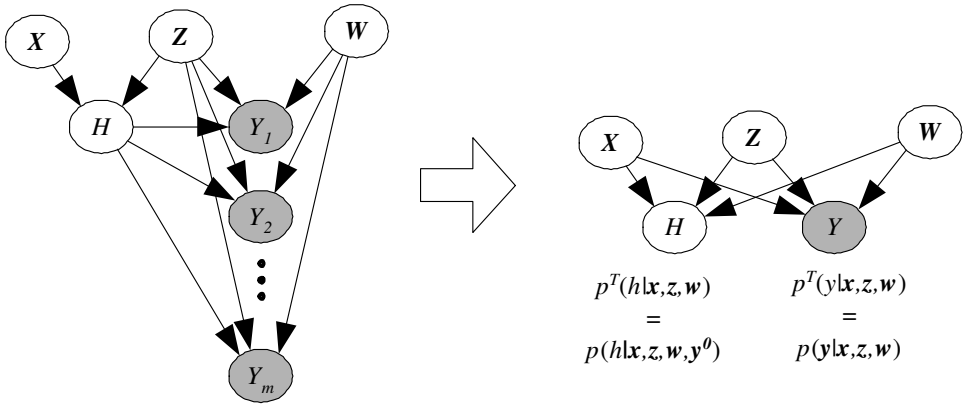


Figure 12: General OCS macro transformation.

B.2.1 THE UNOBSERVED PATH SHORTENING (UPS) ALGORITHM

The unobserved-path-shortening (UPS) algorithm is applied when $\mathcal{G}$ contains only unobserved nodes (all nodes from **O** have been removed before this algorithm is used). We say a node is a *root* if it has no parents. The algorithm takes as input any non-root node Y, and returns the model $\mathcal{G}^T$ in which all nodes have been deleted except for Y and its root-node ancestors **R**. For every $R \in \mathbf{R}$, the edge $R \rightarrow Y$ is in $\mathcal{G}^T$ (the edge need not be in $\mathcal{G}$); $\mathcal{G}^T$ contains no other edges (see Figure 13). In Figure 14, we show how the UPS algorithm is implemented by a sequence of the transformations presented in Section B.1.

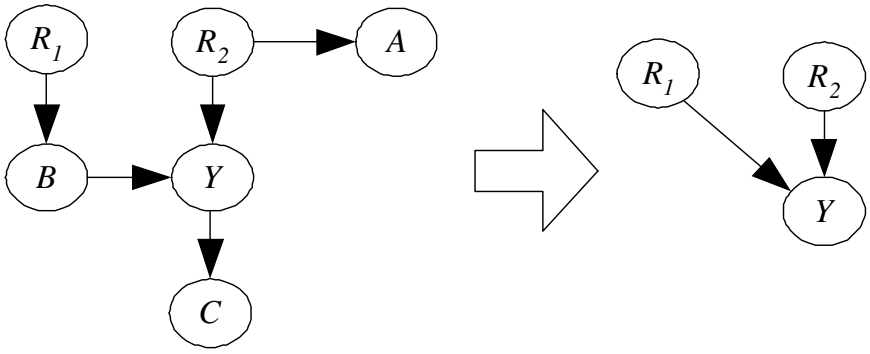


Figure 13: An example application of the UPS algorithm.

The following lemma demonstrates that the steps given in Figure 14 correctly implement the UPS algorithm using graph transformations.

Lemma 20 *Let $\mathcal{G}$ be a Bayesian network containing non-root node Y, and let* **R** *denote the set of root-node ancestors of Y in $\mathcal{G}$. Let $\mathcal{G}^T$ denote the Bayesian network that results from applying Algorithm UPS with inputs $\mathcal{G}$ and Y. Then after the algorithm completes, the nodes in $\mathcal{G}^T$ are precisely $\mathbf{R} \cup \{Y\}$, and the edges in $\mathcal{G}^T$ are precisely $R \rightarrow Y$ for every $R \in \mathbf{R}$.*

Algorithm UPS
Input: Bayesian network $\mathcal{G}$ and non-root node Y. (Let $\mathbf{R}$ denote the set of root-node ancestors of Y in $\mathcal{G}$, and assume that all nodes in $\mathcal{G}$ are unobserved.)
Output: Bayesian network $\mathcal{G}^T$ containing nodes $\mathbf{R} \cup \{Y\}$ and edges $R \rightarrow Y$ for every $R \in \mathbf{R}$

1. Set $\mathcal{G}^T = \mathcal{G}$
2. While $\mathcal{G}^T$ contains at least one barren node $B \neq Y$, delete B using the barren-node removal transformation.
3. While Y has at least one parent that is not a root node
4. Choose any lowest non-root H that is a parent of Y
5. Reverse the edge $H \rightarrow Y$ using the edge-reversal transformation
6. Delete the (now barren) node H using the barren-node-removal transformation.
7. Return $\mathcal{G}^T$

Figure 14: The unobserved path shortening (UPS) algorithm

Proof: First we note that after step 2 of the algorithm, every node in $\mathcal{G}^T$ other than Y is an ancestor of Y. This follows because, given that every node in $\mathcal{G}^T$ is unobserved, any non-ancestor of Y must either be barren or have some descendant (not equal to Y) that is barren.

At step 5, there cannot be any directed path from H to Y other than the edge $H \rightarrow Y$ because H is chosen to be a lowest parent of H. Thus, we know that the edge-reversal transformation at step 5 is always valid. Furthermore, because every node is an ancestor of Y, we know that Y must be the only child of H, and thus after the edge reversal, H must be barren (H cannot gain a child from the reversal), and we can always delete H in step 6.

By definition of an edge reversal, the only nodes that can gain parents in step 5 are Y and H. Because H is necessarily a non-root node, we conclude that every node in $\mathbf{R}$ will remain a root node after every edge reversal. The definition of an edge reversal also guarantees that any node other than H that is an ancestor of Y before the reversal will remain an ancestor after the reversal. Thus when the algorithm terminates, all nodes other than Y must be root-node parents of Y. □

B.2.2 THE OBSERVED NODE ELIMINATION (ONE) ALGORITHM

In this section, we describe the observed-node elimination (ONE) algorithm that deletes all of the observed variables from $\mathcal{G}$ such that, given certain preconditions, the invariant of Equation 5 holds on the resulting model. The details of the algorithm are shown in Figure 15.

In Figure 16, we show an example of the algorithm applied to a model. The original model is shown in Figure 16a, where the observed nodes E, F, and G are depicted by shading. In step 2 of the algorithm, the edges $F \rightarrow C$ and $F \rightarrow G$ are removed, resulting in the model from Figure 16b. For step 3, we see that all four unobserved nodes have at least one observed child. The only lowest nodes are C and D; (arbitrarily) choosing C first (shown with a thick border in the figure), we apply the OCS transformation (which "combines" the singleton node G, covers the edge $C \rightarrow G$ by

Algorithm ONE
Input: Bayesian network $\mathcal{G}$ consisting of observed nodes **O** and unobserved nodes **U**, set of observations **o** corresponding to nodes **O**
Output: Bayesian network $\mathcal{G}^T$ containing only the nodes in **U**

1. Set $\mathcal{G}^T = \mathcal{G}$.
2. For every edge $O \rightarrow X$ in $\mathcal{G}^T$ for which $O \in \mathbf{O}$ is observed, remove the edge using an edge-removal transformation.
3. While there exists an unobserved node in **U** that has at least one child from **O** in $\mathcal{G}^T$, apply the OCS transformation to $\mathcal{G}^T$ using any lowest such node.
4. Delete every node $O \in \mathbf{O}$ by applying the barren-node-elimination transformation.
5. Return $\mathcal{G}^T$.

Figure 15: The observed node elimination (ONE) algorithm

adding $B \rightarrow G$, then reverses and deletes the edge between C and G) resulting in the model shown in Figure 16c. Still in step 3, the lowest nodes are B and D; choosing B for the OCS transformation results in model shown in Figure 16d. In this model, EFG is the combined node of E, F, and G from the OCS transformation. Still in step 3, the lowest nodes are A and D; choosing D for the OCS transformation results in the model shown in Figure 16e. For the last iteration of step 3, A is the only node with an observed child, and applying the OCS transformation results in the model shown in Figure 16f. Finally, in step 4, the barren node EFG is deleted, resulting in the final model shown in Figure 16g that contains only observed nodes.

To help prove properties about Algorithm ONE, we use $\mathcal{G}^i$ to denote the Bayesian network that results after i iterations of the While loop at step 3. We define $\mathcal{G}^0$ to be the graph $\mathcal{G}^T$ that results after applying step 2 but before the first iteration of the While loop at step 3. We use H^i to denote the (lowest) node chosen in iteration i of the While loop, we use $\mathbf{Y}^i$ to denote the set of observed children of H^i on iteration i of the While loop, and we use $Y^i = Comp(\mathbf{Y}^i)$ to denote the composite node created by the OCS transformation in iteration i of the While loop.

As a result of applying the OCS transformation in step 3, we create the new composite node Y^i defined by the subset $\mathbf{Y} \subseteq \mathbf{O}$ of the observed variables. To simplify discussion, we find it convenient to treat **O** as a static set of nodes as opposed to a set that changes each time a new composite node is created. Thus, we will say that any composite node Y^i is contained in the set **O** when technically we should say that all nodes that have been combined to create Y are contained in **O**.

Lemma 21 *For all $\mathcal{G}^i$ (i.e., for all graphs considered in step 3 of Algorithm ONE), every node in* **O** *has zero children.*

Proof: The proposition clearly holds for $\mathcal{G}^0$ due to step 2. Toward a contradiction, let i be the first iteration of the While loop in which a node in **O** gains a child. By definition of the OCS transformation, the only nodes that can gain children are parents of H^i and parents of nodes in $\mathbf{Y}^i$. Because these nodes already have children, they cannot be in **O**. □

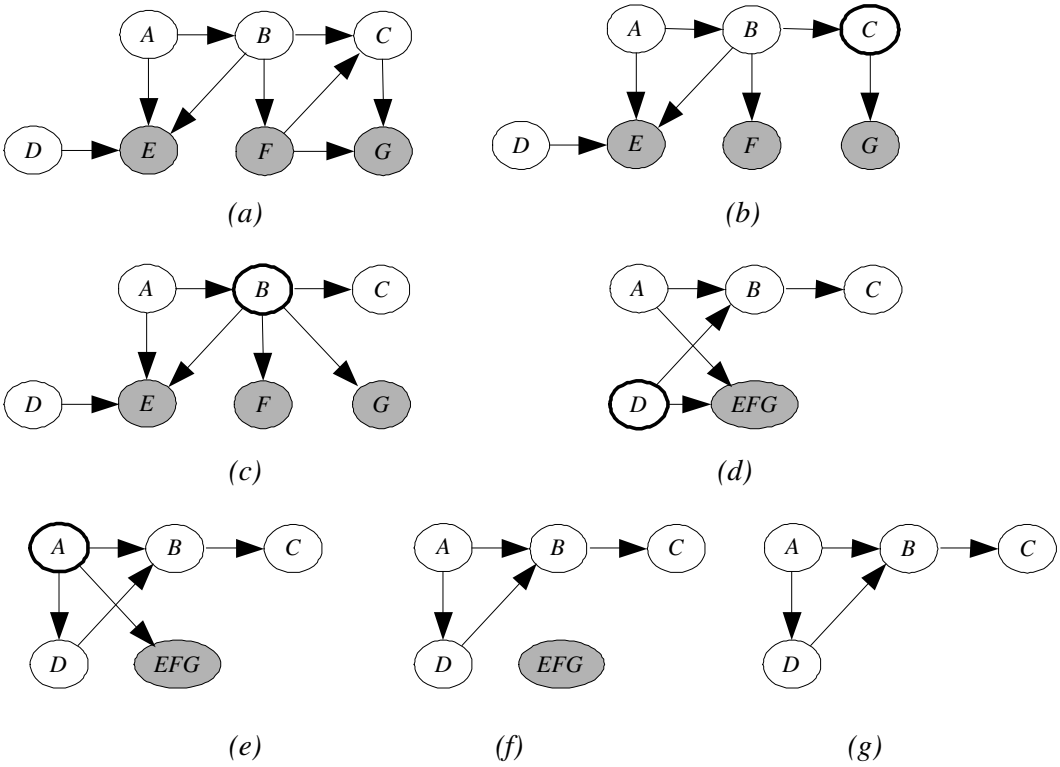


Figure 16: Example of the ONE algorithm applied to a model. Lowest nodes chosen in step 3 are shaded.

Recall from Section B.1.5 that the OCS transformation required that the observed children must have no children themselves. Lemma 21 guarantees that this property holds for every $\mathcal{G}^i$, and thus the OCS transformation can always be applied in step 3. We now demonstrate that the node H^i can be chosen by Algorithm ONE in step 3 at most once. An immediate consequence of this result is that step 3 terminates in at most $|\mathbf{U}|$ iterations.

Proposition 22 *Let $Anc^i(\mathbf{O})$ denote the set of nodes in $\mathbf{U}$ that are ancestors in $\mathcal{G}^i$ of at least one node in $\mathbf{O}$. Then $Anc^i(\mathbf{O}) = Anc^{i-1}(\mathbf{O}) \setminus H^i$.*

Proof: From the definition of the OCS transformation, at each iteration i this "macro" transformation first applies a node-combination transformation on the observed children $\mathbf{Y}^i$ of H^i, then applies an edge-reversal transformation on the edge $H^i \rightarrow Y^i$, and then applies the edge-removal transformation on $Y^i \rightarrow H^i$.

We first note that applying a node-combination transformation on nodes in $\mathbf{O}$ cannot change the set of ancestors of node in $\mathbf{O}$.

In order for a node to become a new ancestor of a node in $\mathbf{O}$, some edge $A \rightarrow B$ must be added to $\mathcal{G}^i$ such that before the addition, A is not an ancestor of a node in $\mathbf{O}$ and B is an ancestor of a node in $\mathbf{O}$. From the definition of the OCS transformation, the only edges that are added are either

(1) of the form $A \rightarrow Y^i$, where A is a parent of H^i and hence an ancestor of node $Y^i \in \mathbf{O}$, or (2) of the form $A \rightarrow H^i$, where A is the parent of $Y^i \in \mathbf{O}$. Thus we conclude that $Anc^i(\mathbf{O}) \subseteq Anc^{i-1}(\mathbf{O})$.

In order for a node to no longer be an ancestor of a node in $\mathbf{O}$, some edge $A \rightarrow B$ must be deleted such that after the deletion, A is no longer an ancestor of $\mathbf{O}$. The only edge that is deleted by the transformation is $H^i \rightarrow Y^i$. After the transformation, all parents of H^i are necessarily parents of Y^i, and thus the only node that can possibly no longer be an ancestor of a node in $\mathbf{O}$ is H^i. Because H^i was chosen as the lowest node with a child in $\mathbf{O}$, and because the only added edges were incident into H^i or a parent of H^i, we know that no descendant of H^i can be an ancestor of a node in $\mathbf{O}$, and the lemma follows. □

Corollary 23 *Algorithm ONE terminates after choosing each node from* $\mathbf{U}$ *in step 3 at most once.*

Proof: As in Lemma 22, we use $Anc^i(\mathbf{O})$ to denote the set of nodes in $\mathbf{U}$ that are ancestors in $\mathcal{G}^i$ of at least one node in $\mathbf{O}$. In order to be chosen in Step 3 during iteration i of the While loop, H^i must be in $Anc^i(\mathbf{O})$. From Lemma 22, if H^i is chosen during iteration i, it can never be an element of $Anc_j(\mathbf{O})$ for $j > i$. □

The next result demonstrates that by breaking ties as appropriate, we can guarantee that any particular unobserved node with no unobserved parents will be a root node after applying Algorithm ONE.

Lemma 24 *Let* $U \in \mathbf{U}$ *be any unobserved node with no unobserved parents in* $\mathcal{G}$*. If in Algorithm ONE we break ties in step 3 in favor of not selecting* U*, then* U *will be a root node in* $\mathcal{G}^T$*.*

Proof: Suppose not. Then at some iteration of the While loop in step 3, an unobserved parent W must be added to U by the algorithm. Let i be the first iteration in which this occurs. By definition of the OCS transformation, we conclude that $H^i = U$ and that W is a parent of Y^i that is not a parent of U. Because we break ties in favor of not choosing U, and because W has a child in $\mathbf{O}$, we conclude that there must be a directed path from W to U. But from Lemma 21 we conclude that the last edge in this directed path is from a node in $\mathbf{U}$ which means that U already has an unobserved parent. □

Appendix C. Properties of Local Distributions

In this appendix, we formally define the BLL and BLTSP properties that were described (informally) in Appendix A, and we show the conditions under which these properties are maintained in the conditional probability distributions as a result of the various graph transformations defined in Appendix B. We begin in Section C.1 by presenting some preliminary definitions and results. In Section C.2, we use this material to derive the main results of this section.

C.1 Preliminary Definitions and Results

In this section, we consider non-negative real-valued functions of $\mathbf{X}$, where $\mathbf{X} = (X_1, \ldots, X_n)$ is a set of variables such that the states of each variable X_i are totally ordered. By *totally ordered*, we mean totally ordered in the *non-strict* sense, where some states may have equal order. For convenience, we often write $f(x_1, \ldots, x_n)$ as $f(\ldots, x_i, \ldots)$ to emphasize the argument x_i.

Definition 25 *The function* $f(\mathbf{X})$ *is* lattice *if, for any* i *such that* $x_i > x_i'$*,*

$$f(\ldots, x_i, \ldots) > f(\ldots, x_i', \ldots),$$

where the other arguments are held fixed and are arbitrary.

For example, for a function with two binary arguments (X_1, X_2) with state orderings $x_1^0 > x_1^1$ and $x_2^0 > x_2^1$, we have $f(x_1^0, x_2^0) > f(x_1^0, x_2^1) > f(x_1^1, x_2^1)$ and $f(x_1^0, x_2^0) > f(x_1^1, x_2^0) > f(x_1^1, x_2^1)$.

Definition 26 *The function $f(\mathbf{X})$ is* totally non-strictly positive *if, for all $i < j$,*

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) \cdot f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) \geq f(\ldots, x_i, \ldots, x_j, \ldots) \cdot f(\ldots, x_i', \ldots, x_j', \ldots). \tag{6}$$

The concept of total non-strict positivity is often referred to as *multivariate total positivity of order two* (see Karlin and Rinott, 1980). We now define a version of total positivity where the inequality in Equation 6 must be strict whenever equality does not hold trivially.

Definition 27 *The function $f(\mathbf{X})$ is* totally strictly positive *if, for all $i < j$,*

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) \cdot f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) \geq f(\ldots, x_i, \ldots, x_j, \ldots) \cdot f(\ldots, x_i', \ldots, x_j', \ldots), \tag{7}$$

where the other arguments are held fixed and are arbitrary, and equality holds if and only if either

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) = f(\ldots, x_i, \ldots, x_j, \ldots) \quad \text{and}$$
$$f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) = f(\ldots, x_i', \ldots, x_j', \ldots)$$

or

$$f(\ldots, \max(x_i, x_i'), \ldots, \max(x_j, x_j'), \ldots) = f(\ldots, x_i', \ldots, x_j', \ldots) \quad \text{and}$$
$$f(\ldots, \min(x_i, x_i'), \ldots, \min(x_j, x_j'), \ldots) = f(\ldots, x_i, \ldots, x_j, \ldots).$$

For example, a function f with two binary arguments (X_1, X_2) with state orderings $x_1^0 > x_1^1$ and $x_2^0 > x_2^1$ is totally strictly positive if $f(x_1^0, x_2^0)\ f(x_1^1, x_2^1) > f(x_1^0, x_2^1)\ f(x_1^1, x_2^0)$. Note that all other combinations of arguments yield trivial equalities. For example, applying the definition when $x_1 = x_1' = x_1^0$, $x_2 = x_2^0$, and $x_2' = x_2^1$, yields the constraint

$$f(\min(x_1^0, x_1^0), \min(x_2^0, x_2^1))\ f(\max(x_1^0, x_1^0), \max(x_2^0, x_2^1)) \geq f(x_1^0, x_2^1)\ f(x_1^0, x_2^0),$$

for which equality holds trivially by solving the left-hand side.

We now give properties of positive, lattice, and totally strictly positive functions whose arguments x are binary with values x^0 and x^1, and with state ordering $x^0 > x^1$. We call functions having only binary arguments *cube functions*. As a shorthand, we use (e.g.) f^{ij} to represent $f(x_1^i, x_2^j)$.

Proposition 28 *Given real numbers $\alpha_1 \geq \alpha_2$ in the interval $(0,1)$ and positive real numbers f^{00}, f^{01}, f^{10}, and f^{11} such that $f^{00} \geq f^{01} \geq f^{11}$ and $f^{00} \geq f^{10}$,*

$$\alpha_1 f^{00} + (1 - \alpha_1) f^{01} \geq \alpha_2 f^{10} + (1 - \alpha_2) f^{11}, \tag{8}$$

where equality holds if and only if $(f^{10} \geq f^{11}) \wedge (f^{00} = f^{10}) \wedge (f^{01} = f^{11}) \wedge ((f^{10} = f^{11}) \vee (\alpha_1 = \alpha_2))$. Equivalently,

$$f^{01} + \alpha_1 (f^{00} - f^{01}) \geq f^{10} + \alpha_2 (f^{10} - f^{11}). \tag{9}$$

Proof: There are two cases to consider.
Case 1: $f^{11} > f^{10}$. Here, the right-hand-side of 8 will be strictly less than $f^{11} \leq f^{01}$. Thus, because the left-hand-side of 8 will be at least f^{01}, 8 holds with strict inequality.
Case 2: $f^{10} \geq f^{11}$. Here, because $f^{00} \geq f^{10}$, $f^{01} \geq f^{11}$, and $0 < \alpha_1 < 1$,

$$\alpha_1 f^{00} + (1-\alpha_1) f^{01} \geq \alpha_1 f^{10} + (1-\alpha_1) f^{11}, \tag{10}$$

where equality holds if and only if $(f^{00} = f^{10}) \wedge (f^{01} = f^{11})$. Because $f^{10} \geq f^{11}$ and $\alpha_1 \geq \alpha_2$, we have

$$\alpha_1 f^{10} + (1-\alpha_1) f^{11} \geq \alpha_2 f^{10} + (1-\alpha_2) f^{11}, \tag{11}$$

where equality holds if and only if $(f^{10} = f^{11}) \vee (\alpha_1 = \alpha_2)$. Inequalities 10 and 11 imply 8, where equality holds if and only if $(f^{00} = f^{10}) \wedge (f^{01} = f^{11}) \wedge ((f^{10} = f^{11} \vee (\alpha_1 = \alpha_2))$. □

Proposition 29 *Given a positive, cube, and lattice function $f(X_1, X_2)$,*

$$\alpha_1 f^{00} + (1-\alpha_1) f^{01} > \alpha_2 f^{10} + (1-\alpha_2) f^{11}, \tag{12}$$

for any two real numbers $0 < \alpha_2 \leq \alpha_1 < 1$.

Proof: Because $f(X_1, X_2)$ is lattice, we have $f^{00} > f^{01} > f^{11}$ and $f^{00} > f^{10} > f^{11}$. The strict inequality 12 therefore follows by Proposition 28. □

Proposition 30 *Given a positive, cube, lattice, and totally strictly positive function $f(X_1, X_2)$,*

$$f^{00} + f^{11} > f^{01} + f^{10}. \tag{13}$$

Proof: We know

$$f^{00} f^{11} > f^{01} f^{10}.$$

Subtracting $f^{01} f^{11}$ from both sides, we obtain

$$(f^{00} - f^{01}) f^{11} > (f^{10} - f^{11}) f^{01}.$$

Because $f^{01} > f^{11}$, we have

$$f^{00} - f^{01} > f^{10} - f^{11}.$$

□

The remainder of this section contains propositions needed to prove the main results in Section C.2. We sometimes use functions having a range of $(0,1)$. We call such functions *unit functions*.

Proposition 31 *Given a real number f in $(0,1)$ and a positive, cube, and totally strictly positive function $g(Y_1, Y_2)$, the ratio*

$$\frac{f g^{00} + (1-f) g^{01}}{f g^{10} + (1-f) g^{11}}$$

is a strictly increasing function of f.

Proof: A straightforward algebraic arrangement yields

$$\frac{fg^{00}+(1-f)g^{01}}{fg^{10}+(1-f)g^{11}} = \frac{g^{01}}{g^{11}} + \frac{1}{1+(1-f)g^{11}/(fg^{10})}\left(\frac{g^{00}}{g^{10}} - \frac{g^{01}}{g^{11}}\right) \tag{14}$$

Because $g(Y_1,Y_2)$ is totally strictly positive, the difference of fractions at the end of Equation 14 is positive. The proposition then follows because g^{11}/g^{10} is positive. □

Proposition 32 *Given a unit, cube, lattice, and totally strictly positive function $f(X_1,X_2)$ and a positive, cube, and lattice function $g(Y)$,*

$$\frac{f^{00}g^0+(1-f^{00})g^1}{f^{01}g^0+(1-f^{01})g^1} > \frac{f^{10}g^0+(1-f^{10})g^1}{f^{11}g^0+(1-f^{11})g^1}. \tag{15}$$

Proof: By Proposition 30, we know that $f^{00}+f^{11} > f^{01}+f^{10}$. Therefore, we have

$$(g^0g^0+g^1g^1)f^{00}f^{11}+g^1(g^0-g^1)(f^{00}+f^{11}) > (g^0g^0+g^1g^1)f^{01}f^{10}+g^1(g^0-g^1)(f^{01}+f^{10}). \tag{16}$$

Adding g^1g^1 to both sides of inequality 16 and factoring, we obtain

$$(f^{00}g^0+(1-f^{00})g^1)(f^{11}g^0+(1-f^{11})g^1) > (f^{01}g^0+(1-f^{01})g^1)(f^{10}g^0+(1-f^{10})g^1).$$

Inequality 15 follows from the fact that terms in the denominator are positive. □

Proposition 33 *Given unit, cube, lattice, and totally strictly positive function $f(X_1,X_2)$ and a positive, cube, and totally strictly positive function $g(Y_1,Y_2)$, where $g(0,Y_2)$ is lattice,*

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{01}g^{10}+(1-f^{01})g^{11}} > \frac{f^{10}g^{00}+(1-f^{10})g^{01}}{f^{11}g^{10}+(1-f^{11})g^{11}}. \tag{17}$$

Proof: The function $g(0,Y_2)$ is unit, cube, and lattice. Consequently, by Proposition 32, we have

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{01}g^{00}+(1-f^{01})g^{01}} > \frac{f^{10}g^{00}+(1-f^{10})g^{01}}{f^{11}g^{00}+(1-f^{11})g^{01}}.$$

Interchanging the denominator on the left-hand-side with the numerator on the right-hand-side, we get

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{10}g^{00}+(1-f^{10})g^{01}} > \frac{f^{01}g^{00}+(1-f^{01})g^{01}}{f^{11}g^{00}+(1-f^{11})g^{01}}. \tag{18}$$

Using Proposition 31 and that fact that $f^{01} > f^{11}$, we obtain

$$\frac{f^{01}g^{00}+(1-f^{01})g^{01}}{f^{01}g^{10}+(1-f^{01})g^{11}} > \frac{f^{11}g^{00}+(1-f^{11})g^{01}}{f^{11}g^{10}+(1-f^{11})g^{11}}$$

or, equivalently,

$$\frac{f^{01}g^{00}+(1-f^{01})g^{01}}{f^{11}g^{00}+(1-f^{11})g^{01}} > \frac{f^{01}g^{10}+(1-f^{01})g^{11}}{f^{11}g^{10}+(1-f^{11})g^{11}}. \tag{19}$$

Inequalities 18 and 19 imply

$$\frac{f^{00}g^{00}+(1-f^{00})g^{01}}{f^{10}g^{00}+(1-f^{10})g^{01}} > \frac{f^{01}g^{10}+(1-f^{01})g^{11}}{f^{11}g^{10}+(1-f^{11})g^{11}}.$$

which is equivalent to 17. □

Proposition 34 *Given a real number f in $(0,1)$ and a positive and cube function $g(Y_1,Y_2,Y_3)$, where $g(0,Y_2,Y_3)$, $g(1,Y_2,Y_3)$, and $g(Y_1,1,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally non-strictly positive,*

$$\frac{fg^{000}+(1-f)g^{001}}{fg^{010}+(1-f)g^{011}} > \frac{fg^{100}+(1-f)g^{101}}{fg^{110}+(1-f)g^{111}}. \tag{20}$$

Proof: Using Equation 14, we rewrite inequality 20 as follows:

$$\begin{aligned}\frac{g^{001}}{g^{011}} + \frac{1}{1+(1-f)g^{011}/(fg^{010})}\left(\frac{g^{000}}{g^{010}}-\frac{g^{001}}{g^{011}}\right) > \\ \frac{g^{101}}{g^{111}} + \frac{1}{1+(1-f)g^{111}/(fg^{110})}\left(\frac{g^{100}}{g^{110}}-\frac{g^{101}}{g^{111}}\right).\end{aligned} \tag{21}$$

Now, observe that inequality 21 has the same form as 9, with

$$\alpha_1 = \frac{1}{1+(1-f)g^{011}/(fg^{010})} \quad \text{and} \quad \alpha_2 = \frac{1}{1+(1-f)g^{111}/(fg^{110})}.$$

In addition, because $g(0,Y_2,Y_3)$ and $g(1,Y_2,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally (non-strictly) positive, we have

$$\frac{g^{000}}{g^{010}} > \frac{g^{001}}{g^{011}} \geq \frac{g^{101}}{g^{111}} \quad \text{and} \quad \frac{g^{000}}{g^{010}} \geq \frac{g^{100}}{g^{110}} > \frac{g^{101}}{g^{111}}.$$

Thus, the conditions of Proposition 28 apply, and 20 will hold if $\alpha_1 > \alpha_2$, or

$$\frac{1}{1+(1-f)g^{011}/(fg^{010})} > \frac{1}{1+(1-f)g^{111}/(fg^{110})}. \tag{22}$$

Rearranging inequality 22 and canceling the terms f and $(1-f)$, we obtain

$$\frac{g^{011}}{g^{010}} < \frac{g^{111}}{g^{110}}$$

which holds because $g(Y_1,1,Y_3)$ is totally strictly positive. □

Proposition 35 *Given a unit, cube, and lattice function $f(X)$ and positive, cube, and lattice function $g(Y_1,Y_2,Y_3)$, where $g(0,Y_2,Y_3)$, $g(1,Y_2,Y_3)$, and $g(Y_1,1,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally (non-strictly) positive,*

$$\frac{f^0g^{000}+(1-f^0)g^{001}}{f^0g^{010}+(1-f^0)g^{011}} > \frac{f^1g^{100}+(1-f^1)g^{101}}{f^1g^{110}+(1-f^1)g^{111}}. \tag{23}$$

Proof: By Proposition 34, we have

$$\frac{f^0g^{000}+(1-f^0)g^{001}}{f^0g^{010}+(1-f^0)g^{011}} > \frac{f^0g^{100}+(1-f^0)g^{101}}{f^0g^{110}+(1-f^0)g^{111}}. \tag{24}$$

Because $f^0 > f^1$ and $g(1,Y_2,Y_3)$ is totally strictly positive, Proposition 31 yields

$$\frac{f^0g^{100}+(1-f^0)g^{101}}{f^0g^{110}+(1-f^0)g^{111}} > \frac{f^1g^{100}+(1-f^1)g^{101}}{f^1g^{110}+(1-f^1)g^{111}}. \tag{25}$$

Inequalities 24 and 25 imply 23. □

Proposition 36 *Given a unit, cube, lattice and totally strictly positive function $f(X_1,X_2)$ and positive, cube, lattice function $g(Y_1,Y_2,Y_3)$, where $g(0,0,Y_3)$ is lattice, $g(0,Y_2,Y_3)$, $g(1,Y_2,Y_3)$, $g(Y_1,0,Y_3)$, and $g(Y_1,1,Y_3)$ are totally strictly positive and $g(Y_1,Y_2,0)$ and $g(Y_1,Y_2,1)$ are totally (non-strictly) positive,*

$$\frac{f^{00}g^{000}+(1-f^{00})g^{001}}{f^{01}g^{010}+(1-f^{01})g^{011}} > \frac{f^{10}g^{100}+(1-f^{10})g^{101}}{f^{11}g^{110}+(1-f^{11})g^{111}}. \tag{26}$$

Proof: Using Proposition 33 and the fact that $g(0,0,Y_3)$ is lattice and $g(Y_1,0,Y_3)$ is totally strictly positive, we get

$$\frac{f^{00}g^{000}+(1-f^{00})g^{001}}{f^{01}g^{000}+(1-f^{01})g^{001}} > \frac{f^{10}g^{100}+(1-f^{10})g^{101}}{f^{11}g^{100}+(1-f^{11})g^{101}}. \tag{27}$$

Because $f(X_1,1)$ is lattice, Proposition 35 yields

$$\frac{f^{01}g^{000}+(1-f^{01})g^{001}}{f^{01}g^{010}+(1-f^{01})g^{011}} > \frac{f^{11}g^{100}+(1-f^{11})g^{101}}{f^{11}g^{110}+(1-f^{11})g^{111}}. \tag{28}$$

Multiplying the left-hand-sizes of inequalities 27 and 28 and the right-hand-sides of the same inequalities, we obtain 26. □

C.2 The BLL and BLTSP Properties

In this section, we turn our attention from general non-negative real-valued functions to conditional probability distributions. In particular, we consider Bayesian networks for discrete, finite-valued variables in which the local distributions $p(y|x_1,\ldots,x_n)$ have the following properties.

Definition 37 *Given a set of variables Y, $X_1,\ldots,X_n$ such that each variable has a finite number of totally ordered states, the distribution $p(y|x_1,\ldots,x_n)$ is* lattice with respect to state y^0 *if the function $f(x_1,\ldots,x_n) = p(y^0|x_1,\ldots,x_n)$ is lattice.*

Definition 38 *Given a set of variables Y, $X_1,\ldots,X_n$ such that each variable has a finite number of totally ordered states, the distribution $p(y|x_1,\ldots,x_n)$ is* totally strictly positive with respect to state y^0 *if the function $f(x_1,\ldots,x_n) = p(y^0|x_1,\ldots,x_n)$ is totally strictly positive.*

We further concentrate on local distributions that are binary-like. In describing such distributions, we need the concept of a *distinguished* state of a variable. For the remainder of the paper, we use x^0 to denote the distinguished state of variable X.

Definition 39 *Given a set of variables Y, $X_1,\ldots,X_n$ such that each variable has a finite number of states and each variable X has a distinguished state x^0, the local distribution $p(y|x_1,\ldots,x_n)$ is* binary-like *if*

$$y \neq y^0 \text{ and } y' \neq y^0 \text{ implies } p(y|x_1,\ldots,x_n) = p(y'|x_1,\ldots,x_n) \tag{29}$$

$$x_i = x_i' \text{ or } (x_i \neq x_i^0 \text{ and } x_i' \neq x_i^0)\ i = 1,\ldots,n \text{ implies } p(y|x_1,\ldots,x_n) = p(y|x_1',\ldots,x_n'). \tag{30}$$

If condition 30 is satisfied for some particular state y, then $p(y|x_1,\ldots,x_n)$ is said to be binary-like with respect to y.

Thus, for a local distribution that is binary-like, if any non-distinguished state is replaced with another non-distinguished state on either side of the conditioning bar, the conditional probability remains the same. For a local distribution that is binary-like with respect to y, if any non-distinguished state on the right-hand side of the conditioning bar is replaced with another non-distinguished state, the conditional probability for state y remains the same. When appropriate, we use x^1 to denote an arbitrary non-distinguished state of X.

When working with distributions that are both binary-like and either lattice or totally strictly positive, we need to be careful how we assign the total ordering to the states for each variable X. In particular, in order for a distribution to be both binary-like and either lattice or totally strictly positive, all non-distinguished states must have equal order in the (non-strict) total ordering. We incorporate this condition in the following definitions. In addition, we use the ordering $x^0 > x^1$ for all variables X.

Definition 40 *Given a set of variables $Y, X_1, \ldots, X_n$ such that each variable has a finite number of totally ordered states, and each variable X has a distinguished state x^0, the distribution $p(y|x_1, \ldots, x_n)$ is* binary-like lattice (BLL) *if (1) the distribution is lattice with respect to y^0, (2) the distribution is binary-like and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order. The distribution $p(y|x_1, \ldots, x_n)$ is* binary-like lattice (BLL) with respect to y^0 *if (1) the distribution is lattice with respect to y^0, (2) the distribution is binary-like with respect to y^0 and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order.*

Definition 41 *Given a set of variables $Y, X_1, \ldots, X_n$ such that each variable has a finite number of totally ordered states, and each variable X has a distinguished state x^0, the distribution $p(y|x_1, \ldots, x_n)$ is* binary-like totally strictly positive (BLTSP) *if (1) the distribution is totally strictly positive with respect to y^0, (2) the distribution is binary-like and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order. The distribution $p(y|x_1, \ldots, x_n)$ is* binary-like totally strictly positive (BLTSP) with respect to y^0 *if (1) the distribution is binary-like with respect to y^0, (2) the distribution is totally strictly positive with respect to y^0, and (3) if, for each variable X, x^0 is greatest in order and all non-distinguished states of X are equal in order.*

In the following sections, we consider the graph transformations of Section B.1, and investigate the conditions under which the BLL and the BLTSP properties are retained in the distributions that result from a transformation. We will say that a node is BLL (BLTSP) to mean that the local distribution for that node is BLL (BLTSP).

C.2.1 BLL AND BLTSP FOR THE EDGE-DELETION TRANSFORMATION

Lemma 42 (Edge Deletion, BLL and BLTSP) *Let $\mathcal{G}$ be a model containing the edge $O \rightarrow U$, where $O \in \mathbf{O}$ is observed, and let $\mathcal{G}^T$ denote the model that results from applying the edge-deletion transformation on $O \rightarrow U$ in $\mathcal{G}$. If U is BLL in $\mathcal{G}$ then U is BLL in $\mathcal{G}^T$, and if U is BLTSP in $\mathcal{G}$ than U is BLTSP in $\mathcal{G}^T$.*

Proof: Let $\mathbf{T}$ be the set of parents of U other than O. From the definition of an edge deletion, $\mathbf{T}$ will be the parents of U in $\mathcal{G}^T$ and, assuming (e.g.) o^* is the observed value of O, we have

$$p^T(u|\mathbf{t}) = p(u|\mathbf{t}, o^*)$$

for all u and $\mathbf{t}$. From the definition of BLL, if $p(u|\mathbf{t},o)$ is BLL, then it is also BLL when restricted to $o = o^*$. Similarly, from the definition of BLTSP, if $p(u|\mathbf{t},o)$ is BLTSP, then it is also BLTSP when restricted to $o = o^*$. □

C.2.2 BLL AND BLTSP FOR THE BARREN-NODE-REMOVAL TRANSFORMATION

Proposition 43 *Let $\mathcal{G}$ be a model containing barren node X, and let $\mathcal{G}^T$ denote the model that results from applying a barren-node-removal transformation to $\mathcal{G}$ on X. For any node $Y \neq X$ in $\mathcal{G}$, we have: (1) if Y is BLL in $\mathcal{G}$, then Y is BLL in $\mathcal{G}^T$, and (2) if Y is BLTSP in $\mathcal{G}$, then Y is BLTSP in $\mathcal{G}^T$.*

Proof: Follows immediately from the definition of a barren-node removal because the local distributions that remain in $\mathcal{G}^T$ are identical in $\mathcal{G}$. □

C.2.3 BLL AND BLTSP FOR THE OCS TRANSFORMATION

Lemma 44 (OCS, BLL in Y) *Consider the OCS transformation shown in Figure 12, where $p(h|\mathbf{x},\mathbf{z})$ is BLL. If $p(y_j|\mathbf{z},\mathbf{w},h)$ is BLL with respect to y_j^0, $j = 1,\ldots,m$, then $p^T(y|\mathbf{x},\mathbf{z},\mathbf{w}) = p(\mathbf{y}|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLL with respect to $y^0 = \mathbf{y}^0$. If $m = 1$ and $p(\mathbf{y}|\mathbf{z},\mathbf{w},h) = p(y_1|\mathbf{z},\mathbf{w},h)$ is BLL, then $p^T(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLL.*

Proof: For notational simplicity, we use Y to denote the set of Y_j nodes in the original graph $\mathcal{G}$; that is, we use the node Y from $\mathcal{G}^T$ as shorthand for the set of nodes that were combined to create that variable.

First, we show that $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is either binary-like or binary-like with respect to y^0. From the sum rule of probability, we have

$$p(y|\mathbf{x},\mathbf{z},\mathbf{w}) = p(h^0|\mathbf{x},\mathbf{z})\ p(y|\mathbf{z},\mathbf{w},h^0) + \sum_{h \neq h^0} p(h|\mathbf{x},\mathbf{z})\ p(y|\mathbf{z},\mathbf{w},h). \tag{31}$$

When $m = 1$ and $p(y|\mathbf{z},\mathbf{w},h)$ is binary-like, because $p(h|\mathbf{x},\mathbf{z})$ is also binary like, we can rewrite Equation 31 as follows:

$$p(y|\mathbf{x},\mathbf{z},\mathbf{w}) = p(h^0|\mathbf{x},\mathbf{z})\ p(y|\mathbf{z},\mathbf{w},h^0) + (1 - p(h^0|\mathbf{x},\mathbf{z}))\ p(y|\mathbf{z},\mathbf{w},h^1). \tag{32}$$

Because $p(h|\mathbf{x},\mathbf{z})$ is binary-like with respect to h^0 and the remaining two terms in Equation 32 are binary-like, it follows that $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like. When $m \geq 1$ and $p(y_j|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y_j^0, $j = 1,\ldots,m$, Equation 32 with $Y = y^0$ still holds, because $p(y|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y^0. Consequently, $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like with respect to y^0. It remains to show that $p(y^0|\mathbf{x},\mathbf{z},\mathbf{w})$ is lattice. There are three cases to consider.

Case 1: $X \in \mathbf{X}$ changes. If $\mathbf{X}$ is empty, there is nothing to prove, so assume $\mathbf{X}$ is not empty. Here, we need to show that $p(y^0|x^0,\mathbf{x}',\mathbf{z},\mathbf{w}) > p(y^0|x^1,\mathbf{x}',\mathbf{z},\mathbf{w})$, where $\mathbf{X}' = \mathbf{X} \setminus \{X\}$. Using Equation 32 with $Y = y^0$ and omitting those variables that are held constant, we rewrite this condition as

$$\begin{aligned} p(h^0|x^0)\ p(y^0|h^0) + (1 - p(h^0|x^0))\ p(y^0|h^1) > \\ p(h^0|x^1)\ p(y^0|h^0) + (1 - p(h^0|x^1))\ p(y^0|h^1). \end{aligned} \tag{33}$$

Because $p(y_j^0|h)$ is lattice, $j = 1,\ldots,m$, we know that $p(y^0|h^i) = \prod_{j=1}^m p(y_j^0|h^i)$ is lattice—that is, $p(y^0|h^0) > p(y^0|h^1)$. Because $p(h^0|x)$, we have $p(h^0|x^0) > p(h^0|x^1)$. Consequently, inequality 33 holds.

Case 2: $W \in \mathbf{W}$ changes. If $\mathbf{W}$ is empty there is nothing to prove, so assume $\mathbf{W}$ is not empty. Here, we need to show that $p(y^0|\mathbf{x},\mathbf{z},w^0,\mathbf{w}') > p(y^0|\mathbf{x},\mathbf{z},w^1,\mathbf{w}')$, where $\mathbf{W}' = \mathbf{W} \setminus \{W\}$. Again using Equation 32 and omitting those variables that are held constant, this condition becomes

$$\begin{aligned} &p(h^0)\ p(y^0|w^0,h^0) + (1-p(h^0))\ p(y^0|w^0,h^1) > \\ &\quad p(h^0)\ p(y^0|w^1,h^0) + (1-p(h^0))\ p(y^0|w^1,h^1). \end{aligned} \tag{34}$$

First note that $p(y^0|w,h)$ is lattice. To see this fact, write

$$p(y^0|w,h) = \prod_a p(y_a^0|w,h) \prod_b p(y_b^0|h),$$

where each Y_a has parents W and H and each Y_b has parent H. Because there is at least one Y_a and $p(y_a^0|w,h)$ is lattice, it follows that $p(y^0|w,h)$ is lattice. Identifying $p(h^0)$ with $\alpha_1 = \alpha_2$ and $p(y^0|w^i,h^j)$ with f^{ij} in Proposition 29, and noting that f^{ij} is lattice because $p(y^0|w,h)$ is lattice, we find that inequality 34 holds.

Case 3: $Z \in \mathbf{Z}$ changes. If $\mathbf{Z}$ is empty there is nothing to prove, so assume $\mathbf{Z}$ is not empty. Here, we need to show that $p(y^0|\mathbf{x},z^0,\mathbf{z}',\mathbf{w}) > p(y^0|\mathbf{x},z^1,\mathbf{z}',\mathbf{w})$, where $\mathbf{Z}' = \mathbf{Z} \setminus \{Z\}$. Using Equation 32, this condition becomes

$$\begin{aligned} &p(h^0|z^0)\ p(y^0|z^0,h^0) + (1-p(h^0|z^0))\ p(y^0|z^0,h^1) > \\ &\quad p(h^0|z^1)\ p(y^0|z^1,h^0) + (1-p(h^0|z^1))\ p(y^0|z^1,h^1). \end{aligned} \tag{35}$$

By an argument analogous to that in Case 2 of this Lemma, it follows that $p(y^0|z,h)$ is lattice. Thus, identifying $p(h^0|z^i)$ with α_i and $p(y^0|z^i,h^j)$ with f^{ij} in Proposition 29 and using the fact that $p(h^0|z)$ and $p(y^0|z,h)$ are lattice, we establish inequality 35. □

Lemma 45 (OCS, BLL in H) *Consider the OCS transformation shown in Figure 12, where $p(h|\mathbf{x},\mathbf{z})$ is BLL. If $p(y_j|\mathbf{z},\mathbf{w},h)$ is BLTSP with respect to y_j^0, $j = 1,\ldots,m$, then $p^T(h|\mathbf{x},\mathbf{z},\mathbf{w}) = p(h|\mathbf{y}^0,\mathbf{x},\mathbf{z},\mathbf{w})$ is BLL.*

Proof: As in the proof of Lemma 44, we use Y to denote the set of Y_j nodes in the original graph $\mathcal{G}$.

From Bayes' rule and the definition of Y, we have

$$p(h|y^0,\mathbf{x},\mathbf{z},\mathbf{w}) = \frac{p(h|\mathbf{x},\mathbf{z})\ p(y^0|\mathbf{z},\mathbf{w},h)}{p(h^0|\mathbf{x},\mathbf{z})\ p(y^0|\mathbf{z},\mathbf{w},h^0) + (1-p(h^0|\mathbf{x},\mathbf{z}))\ p(y^0|\mathbf{z},\mathbf{w},h^1)}. \tag{36}$$

where $p(y^0|\mathbf{z},\mathbf{w},h) = \prod_{j=1}^m p(y_j^0|\mathbf{z},\mathbf{w},h)$. Because $p(h|\mathbf{x},\mathbf{z})$ is binary-like and $p(y_j|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y_j^0, $i = 1,\ldots,m$, it follows that $p(h|y^0,\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like. It remains to show that $p(h^0|y^0,\mathbf{x},\mathbf{z},\mathbf{w})$ is lattice.

Dividing the numerator and denominator of the right-hand-side of Equation 36 by the numerator and setting h to h^0, we obtain

$$p(h^0|y^0,\mathbf{x},\mathbf{z},\mathbf{w}) = \frac{1}{1 + \frac{(1-p(h^0|\mathbf{x},\mathbf{z}))}{p(h^0|\mathbf{x},\mathbf{z})} \frac{p(y^0|\mathbf{z},\mathbf{w},h^1)}{p(y^0|\mathbf{z},\mathbf{w},h^0)}}. \tag{37}$$

There are three cases to consider.

Case 1: $X \in \mathbf{X}$ changes. If $\mathbf{X}$ is empty, there is nothing to prove, so assume $\mathbf{X}$ is not empty. Here, we need to show that $p(h^0|y^0,x^0,\mathbf{x}',\mathbf{z},\mathbf{w}) > p(h^0|y^0,x^1,\mathbf{x}',\mathbf{z},\mathbf{w})$, where $\mathbf{X}' = \mathbf{X} \setminus \{X\}$. Using Equation 37 and the fact that $1/(1+a) > 1/(1+b)$ if and only if $1/a > 1/b$ for positive a and b, this condition becomes

$$\frac{p(h^0|x^0)}{(1-p(h^0|x^0))}\frac{p(y^0|h^0)}{p(y^0|h^1)} > \frac{p(h^0|x^1)}{(1-p(h^0|x^1))}\frac{p(y^0|h^0)}{p(y^0|h^1)},$$

which holds because $p(h^0|x)$ is lattice.

Case 2: $W \in \mathbf{W}$ changes. If $\mathbf{W}$ is empty, there is nothing to prove, so assume $\mathbf{W}$ is not empty. Here, we need to show that $p(h^0|y^0,\mathbf{x},\mathbf{z},w^0,\mathbf{w}') > p(h^0|y^0,\mathbf{x},\mathbf{z},w^1,\mathbf{w}')$, where $\mathbf{W}' = \mathbf{W} \setminus \{W\}$. By an argument similar to that in Case 1, this condition becomes

$$\frac{p(h^0)}{(1-p(h^0))}\frac{p(y^0|w^0,h^0)}{p(y^0|w^0,h^1)} > \frac{p(h^0)}{(1-p(h^0))}\frac{p(y^0|w^1,h^0)}{p(y^0|w^1,h^1)}.$$

Canceling the terms involving $p(h^0)$, we see that this inequality holds if $p(y^0|h,w)$ is totally strictly positive. To establish the latter fact, recall that

$$p(y^0|w,h) = \prod_a p(y_a^0|w,h)\prod_b p(y_b^0|h),$$

where each Y_a has parents W and H and each Y_b has parent H. Now note that the product of two functions that are positive and totally strictly positive is also positive and totally strictly positive, and that if $f(X_1,X_2)$ is positive and totally strictly positive and $g(X_1)$ is positive, then $f(X_1,X_2)\cdot g(X_1)$ is positive and totally strictly positive.

Case 3: $Z \in \mathbf{Z}$ changes. If $\mathbf{Z}$ is empty, there is nothing to prove, so assume $\mathbf{X}$ is not empty. Here, we need to show that $p(h^0|y^0,\mathbf{x},z^0,\mathbf{z},\mathbf{w}) > p(h^0|y^0,\mathbf{x},z^1,\mathbf{z},\mathbf{w})$, where $\mathbf{Z}' = \mathbf{Z} \setminus \{Z\}$. This condition becomes

$$\frac{p(h^0|z^0)}{(1-p(h^0|z^0))}\frac{p(y^0|z^0,h^0)}{p(y^0|z^0,h^1)} > \frac{p(h^0|z^1)}{(1-p(h^0|z^1))}\frac{p(y^0|z^1,h^0)}{p(y^0|z^1,h^1)}. \tag{38}$$

By an argument analogous to the last one in Case 2, $p(y^0|z,h)$ is totally strictly positive. Also, $p(h^0|x)$ is lattice. The inequality therefore holds. □

In the base case of the proof of our main result (Theorem 53), we require only that the distributions have the BLL property. The proof of the preceding lemma shows why BLTSP is a required property in the original model. Namely, in Case 2, $p(h^0|y^0,w)$ is lattice if and only if $p(y^0|w,h)$ is totally strictly positive.

Lemma 46 (OCS, BLTSP in Y) *Consider the OCS transformation shown in Figure 12, where $p(h|\mathbf{x},\mathbf{z})$ is BLL and BLTSP. If $p(y_j|\mathbf{z},\mathbf{w},h)$ is BLL with respect to y_j^0 and BLTSP with respect to y_j^0, $j = 1,\ldots,m$, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLTSP with respect to y^0. If $m = 1$ and $p(y|\mathbf{z},\mathbf{w},h) = p(y_1|\mathbf{z},\mathbf{w},h)$ is BLL and BLTSP, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is BLTSP.*

Proof: In the proof of Lemma 44, we showed that (1) if $m = 1$ and $p(y|\mathbf{z},\mathbf{w},h)$ is binary-like, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like; and (2) if $m \geq 1$ and $p(y_j|\mathbf{z},\mathbf{w},h)$ is binary-like with respect to y_j^0, $j = 1,\ldots,m$, then $p(y|\mathbf{x},\mathbf{z},\mathbf{w})$ is binary-like with respect to y^0. It remains to show that $p(y^0|\mathbf{x},\mathbf{z},\mathbf{w})$ is totally strictly positive. There are six cases to consider.

Case 1: $X_1, X_2 \in \mathbf{X}$ changes. Assume $|\mathbf{X}| \geq 2$. Here, we need to show that

$$\frac{p(y^0|x_1^0, x_2^0, \mathbf{x}', \mathbf{z}, \mathbf{w})}{p(y^0|x_1^0, x_2^1, \mathbf{x}', \mathbf{z}, \mathbf{w})} > \frac{p(y^0|x_1^1, x_2^0, \mathbf{x}', \mathbf{z}, \mathbf{w})}{p(y^0|x_1^1, x_2^1, \mathbf{x}', \mathbf{z}, \mathbf{w})}. \tag{39}$$

where $\mathbf{X}' = \mathbf{X} \setminus \{X_1, X_2\}$. Using Equation 32 for $Y = y^0$ and omitting those variables that are held constant, we rewrite this condition as

$$\begin{aligned} &\frac{p(h^0|x_1^0, x_2^0)\ p(y^0|h^0) + (1 - p(h^0|x_1^0, x_2^0))\ p(y^0|h^1)}{p(h^0|x_1^0, x_2^1)\ p(y^0|h^0) + (1 - p(h^0|x_1^0, x_2^1))\ p(y^0|h^1)} > \\ &\quad \frac{p(h^0|x_1^1, x_2^0)\ p(y^0|h^0) + (1 - p(h^0|x_1^1, x_2^0))\ p(y^0|h^1)}{p(h^0|x_1^1, x_2^1)\ p(y^0|h^0) + (1 - p(h^0|x_1^1, x_2^1))\ p(y^0|h^1)}. \end{aligned} \tag{40}$$

Because $p(y_j^0|h)$ is lattice, $j = 1, \ldots, m$, we know that $p(y^0|h) = \prod_{j=1}^m p(y_j^0|h)$ is lattice. Thus, identifying $p(h^0|x_1^i, x_2^j)$ with f^{ij} and $p(y^0|h^i)$ with g^i in Proposition 32, we find that inequality 40 holds.

Case 2: $X \in \mathbf{X}$ and $W \in \mathbf{W}$ changes. Assume $|\mathbf{X}| \geq 1$ and $|\mathbf{W}| \geq 1$. Using Equation 32, we need to show the inequality

$$\begin{aligned} &\frac{p(h^0|x^0)\ p(y^0|w^0, h^0) + (1 - p(h^0|x^0))\ p(y^0|w^0, h^1)}{p(h^0|x^0)\ p(y^0|w^1, h^0) + (1 - p(h^0|x^0))\ p(y^0|w^1, h^1)} > \\ &\quad \frac{p(h^0|x^1)\ p(y^0|w^0, h^0) + (1 - p(h^0|x^1))\ p(y^0|w^0, h^1)}{p(h^0|x^1)\ p(y^0|w^1, h^0) + (1 - p(h^0|x^1))\ p(y^0|w^1, h^1)}. \end{aligned} \tag{41}$$

By an argument analogous to one in Case 2 of Lemma 45, we know that $p(y^0|w, h)$ is totally strictly positive. Therefore, identifying $p(y^0|w^i, h^j)$ with g^{ij} in Proposition 31 and noting that $p(h^0|x^0) > p(h^0|x^1)$, we establish inequality 41.

Case 3: $X \in \mathbf{X}$ and $Z \in \mathbf{Z}$ changes. Assume $|\mathbf{X}| \geq 1$ and $|\mathbf{Z}| \geq 1$. Using Equation 32, we need to show the inequality

$$\begin{aligned} &\frac{p(h^0|x^0, z^0)\ p(y^0|z^0, h^0) + (1 - p(h^0|x^0, z^0))\ p(y^0|z^0, h^1)}{p(h^0|x^0, z^1)\ p(y^0|z^1, h^0) + (1 - p(h^0|x^0, z^1))\ p(y^0|z^1, h^1)} > \\ &\quad \frac{p(h^0|x^1, z^0)\ p(y^0|z^0, h^0) + (1 - p(h^0|x^1, z^0))\ p(y^0|z^0, h^1)}{p(h^0|x^1, z^1)\ p(y^0|z^1, h^0) + (1 - p(h^0|x^1, z^1))\ p(y^0|z^1, h^1)}. \end{aligned}$$

Because $p(y^0|z^0, h) = \prod_{j=1}^m p(y_j^0|z^0, h)$, we know that $p(y^0|z^0, h)$ is lattice. By an argument analogous to one in Case 2 of Lemma 45, we know that $p(y^0|z, h)$ is totally strictly positive. Identifying $p(h^0|x^i, z^j)$ with f^{ij} and $p(y^0|z^i, h^j)$ with g^{ij}, Proposition 33 establishes this identity.

Case 4: $W_1, W_2 \in \mathbf{W}$ changes. Assume $|\mathbf{W}| \geq 2$. Using Equation 32, we need to show the inequality

$$\begin{aligned} &\frac{p(h^0)\ p(y^0|w_1^0, w_2^0, h^0) + (1 - p(h^0))\ p(y^0|w_1^0, w_2^0, h^1)}{p(h^0)\ p(y^0|w_1^0, w_2^1, h^0) + (1 - p(h^0))\ p(y^0|w_1^0, w_2^1, h^1)} > \\ &\quad \frac{p(h^0)\ p(y^0|w_1^1, w_2^0, h^0) + (1 - p(h^0))\ p(y^0|w_1^1, w_2^0, h^1)}{p(h^0)\ p(y^0|w_1^1, w_2^1, h^0) + (1 - p(h^0))\ p(y^0|w_1^1, w_2^1, h^1)}. \end{aligned}$$

Identifying $p(h^0)$ with f and $p(y^0|w_1^i, w_2^j, h^k)$ with g^{ijk}, Proposition 34 establishes this identity if we can show that g^{ijk} satisfies the strict and non-strict total positivity conditions of the proposition. To do so, write

$$p(y^0|w_1,w_2,h) = \prod_a p(y_a^0|w_1,w_2,h) \prod_b p(y_b^0|w_1,h) \prod_c p(y_c^0|w_2,h) \prod_d p(y_c^0|h),$$

where each Y_a has parents W_1, W_2, and H, each Y_b has parents W_1 and H, each Y_c has parents W_2 and H, and each Y_d has parent H. It is not difficult to show that, if there is at least one variable having both W_1 and W_2 as parents, then the product is totally strictly positive. If there is no such variable, however, the product is not totally strictly positive, because $g000/g010 = g100/g110$ and $g001/g011 = g101/g111$. Nonetheless, it is not difficult to show that the remaining four pairwise total strict positivity conditions hold. Consequently, the conditions of Proposition 34 hold.

Case 5: $Z \in \mathbf{Z}$ and $W \in \mathbf{W}$ changes. Assume $|\mathbf{Z}| \geq 1$ and $|\mathbf{W}| \geq 1$. Using Equation 32, we need to show the inequality

$$\frac{p(h^0|z^0)\ p(y^0|z^0,w^0,h^0) + (1-p(h^0|z^0))\ p(y^0|z^0,w^0,h^1)}{p(h^0|z^0)\ p(y^0|z^0,w^1,h^0) + (1-p(h^0|z^0))\ p(y^0|z^0,w^1,h^1)} > \frac{p(h^0|z^1)\ p(y^0|z^1,w^0,h^0) + (1-p(h^0|z^1))\ p(y^0|z^1,w^0,h^1)}{p(h^0|z^1)\ p(y^0|z^1,w^1,h^0) + (1-p(h^0|z^1))\ p(y^0|z^1,w^1,h^1)}.$$

Identifying $p(h^0|z^i)$ with f^i and $p(y^0|z^i, w^j, h^k)$ with g^{ijk}, Proposition 35 establishes this identity if g^{ijk} satisfies the strict and non-strict total positivity conditions of the Proposition. By an argument analogous to one in Case 4 of this Lemma, g^{ijk} satisfies these conditions.

Case 6: $Z_1, Z_2 \in \mathbf{Z}$ changes. Assume $|\mathbf{Z}| \geq 2$. Using Equation 32, we need to show the inequality

$$\frac{p(h^0|z_1^0,z_2^0)\ p(y^0|z_1^0,z_2^0,h^0) + (1-p(h^0|z_1^0,z_2^0))\ p(y^0|z_1^0,z_2^0,h^1)}{p(h^0|z_1^0,z_2^1)\ p(y^0|z_1^0,z_2^1,h^0) + (1-p(h^0|z_1^0,z_2^1))\ p(y^0|z_1^0,z_2^1,h^1)} > \frac{p(h^0|z_1^1,z_2^0)\ p(y^0|z_1^1,z_2^0,h^0) + (1-p(h^0|z_1^1,z_2^0))\ p(y^0|z_1^1,z_2^0,h^1)}{p(h^0|z_1^1,z_2^1)\ p(y^0|z_1^1,z_2^1,h^0) + (1-p(h^0|z_1^1,z_2^1))\ p(y^0|z_1^1,z_2^1,h^1)}.$$

Identifying $p(h^0|z_1^i, z_2^j)$ with f^{ij} and $p(y^0|z_1^i, z_2^j, h^k)$ with g^{ijk}, Proposition 36 establishes this identity if g^{ijk} satisfies the conditions of the Proposition. Because $p(y^0|z_1^0, x_2^0, h) = \prod_{j=1}^m p(y_j^0|z_1^0, z_2^0, h)$, we know that $p(y^0|z_1^0, z_2^0, h)$ is lattice. By an argument analogous to that of Case 4 of this Lemma, g^{ijk} satisfies the strict and non-strict total positivity conditions of Proposition 36. □

Putting the three previous lemmas together, we get the following general result for the OCS transformation.

Corollary 47 (OCS transformation) *Let $\mathcal{G}$ be a Bayesian network, and let $\mathcal{G}^T$ be the result of applying the observed-child-separation transformation on unobserved node H with observed children $\mathbf{Y}$. If the observed values for the children $\mathbf{Y}$ are the distinguished states $\mathbf{y}^0$ for those children, and if in $\mathcal{G}$, H is both BLL and BLTSP, and each observed child $Y_i \in \mathbf{Y}$ is both BLL with respect to y_i^0 and BLTSP with respect to y_i^0, then in $\mathcal{G}^T$, (1) H is BLL, (2) $Y = Comp(\mathbf{Y})$ is BLL with respect to y^0, and (3) $Y = Comp(\mathbf{Y})$ BLTSP with respect to y^0.*

Proof: (1), (2), and (3) follow immediately from Lemma 45, Lemma 44, and Lemma 46, respectively. □

C.2.4 BLL AND BLTSP FOR THE EDGE-REVERSAL TRANSFORMATION

Corollary 48 (Edge Reversal, BLL in Y) *Consider the edge-reversal transformation shown in Figure 8. If the local distributions for H and Y are BLL in $\mathcal{G}$, then the local distribution for Y in $\mathcal{G}^T$ is BLL.*

Proof: The local distribution for Y after reversing $H \rightarrow Y$ is exactly the same as the local distribution for Y after a one-child OCS transformation for $H \rightarrow Y$; thus, the corollary follows from the $m = 1$ case of Lemma 44. □

Corollary 49 (Edge Reversal, BLTSP in Y) *Consider the edge-reversal transformation shown in Figure 8. If the local distribution for H is BLL and BLTSP in $\mathcal{G}$, and if the local distribution for Y is BLL and BLTSP in $\mathcal{G}$, then the local distribution for Y in $\mathcal{G}^T$ is BLTSP.*

Proof: The local distribution for Y after reversing $H \rightarrow Y$ is exactly the same as the local distribution for Y after a one-child OCS transformation for $H \rightarrow Y$; thus, the corollary follows from the $m = 1$ case of Lemma 46. □

C.2.5 BLL FOR THE UPS ALGORITHM

We now show that if every node in $\mathcal{G}$ is BLL, then every node in the graph that results from applying the UPS algorithm is also BLL.

Lemma 50 *Let $\mathcal{G}$ be any Bayesian network, and let $\mathcal{G}^T$ be the result of applying the UPS algorithm with Bayesian network $\mathcal{G}$ and non-root node Y. If every node in $\mathcal{G}$ is BLL, then every node in $\mathcal{G}^T$ is BLL.*

Proof: The result follows because, from Corollary 48, after each edge $H \rightarrow Y$ is reversed, Y remains BLL; the property need not hold for H after the reversal because H is immediately removed. □

C.2.6 BLL FOR THE ONE ALGORITHM

Recall Algorithm ONE from Section B.2.2 which eliminates all observed nodes from a model. In this section, we show conditions under which the nodes that remain after the algorithm are BLL. As in Section B.2.2, we use $\mathcal{G}^i$ to denote the Bayesian network that results after i iterations of the While loop at step 3 of the algorithm, we define $\mathcal{G}^0$ to be the graph $\mathcal{G}^T$ that results after applying step 2 but before the first iteration of the While loop at step 3, we use H^i to denote the (lowest) node chosen in iteration i of the While loop, we use $\mathbf{Y}^i$ to denote the set of observed children of H^i on iteration i of the While loop, and we use $Y^i = Comp(\mathbf{Y}^i)$ to denote the composite node created by the OCS transformation in iteration i of the While loop.

Lemma 51 *Let $\mathcal{G}$ be a Bayesian network in which all nodes are both BLL and BLTSP, and let $\mathbf{o}$ be a set of observations for nodes $\mathbf{O}$. If for every $O \in \mathbf{O}$ that has at least one parent not in $\mathbf{O}$, the observation in $\mathbf{o}$ for O is the distinguished state o^0, then every node in the Bayesian network $\mathcal{G}^T$ that results from applying Algorithm ONE to $\mathcal{G}$ is BLL.*

Proof: From Lemma 42, we know that we retain the BLL and BLTSP properties of all nodes while removing edges in step 2, and thus all nodes in $\mathcal{G}^0$ are both BLL and BLTSP. Similarly,

from Proposition 43, step 4 does not affect the BLL or BLTSP properties of the nodes that remain. Thus, the lemma follows if we can show that for every OCS transformation applied in step 3 of the algorithm, every non-observed node retains the BLL property.

Consider the *ith* iteration of the While loop in step 3. From Corollary 47, if (1) the observed state for each child in $\mathbf{Y}^i$ is the distinguished state for that child, (2) H^i is both BLL and BLTSP, and (3) each child in $\mathbf{Y}^i$ is both BLL with respect to its distinguished state and BLTSP with respect to its distinguished state, then we are guaranteed that all observed variables retain the BLL property. We now demonstrate that these three preconditions of Corollary 47 hold for every iteration *i*.

After applying step 2 of the algorithm, any observed node without at least one observed parent will be completely disconnected from the graph, and thus precondition (1) is always satisfied. From Corollary 23, each unobserved node is chosen at most once in step 3. Because the parents (and hence the local distribution) for an unobserved node only change when it is chosen in step 3, we conclude that precondition (2) is always satisfied.

The only local distributions for nodes in $\mathbf{O}$ that change in iteration *i* are the nodes $\mathbf{Y}^i$, which are replaced by the single node Y^i. From Corollary 47, if preconditions (1) and (2) hold, and if every node $O \in \mathbf{O}$ is both BLL with respect to o^0 and BLTSP with respect to o^0 *before* the transformation, then every node $O \in \mathbf{O}$ is both BLL with respect to o^0 and BLTSP with respect to o^0 *after* the transformation. Because all nodes in $\mathbf{O}$ are initially both BLL with respect to their distinguished states and BLTSP with respect to their distinguished states, precondition (3) always holds and the lemma follows. □

Appendix D. Main Results

In this appendix, we prove Lemma 17 and Lemma 5 using results established in the previous appendices.

Lemma 52 *Let $\mathcal{G}$ be any Bayesian network in which all local distributions are BLL, and let X be any root node in $\mathcal{G}$. If X and Y are d-connected by an $\emptyset$-active path in $\mathcal{G}$, then $p(y^0|x^0) > p(y^0|x^1)$.*

Proof: From Lemma 50, we can apply Algorithm UPS to $\mathcal{G}$ and node Y, and every node in the resulting model $\mathcal{G}^T$ will be BLL. Furthermore, we know from Lemma 20 that X is a root-node parent of Y, and that all other nodes $\mathbf{Z}$ in $\mathcal{G}^T$ are also root-node parents of Y. Expressing the difference of interest using $\mathcal{G}^T$:

$$p(y^0|x^0) - p(y^0|x^1) = \left[\sum_{\mathbf{Z}} p^T(y^0|x^0,\mathbf{z})p^T(\mathbf{z}|x^0)\right] - \left[\sum_{\mathbf{Z}} p^T(y^0|x^1,\mathbf{z})p^T(\mathbf{z}|x^0)\right].$$

Because all nodes in $\mathbf{Z}$ are d-separated from X in $\mathcal{G}^T$ whenever Y is not in the conditioning set we have

$$p(y^0|x^0) - p(y^0|x^1) = \sum_{\mathbf{Z}} p(\mathbf{z}) \left[p^T(y^0|x^0,\mathbf{z}) - p^T(y^0|x^1,\mathbf{z})\right].$$

Every difference in the sum above is guaranteed to be greater than zero by definition of BLL. □

Theorem 53 *Let $\mathcal{G}$ be a Bayesian network in which all conditional distributions are BLL and BLTSP, and let $\mathbf{o}$ be a set of observations for nodes $\mathbf{O}$. If for every $O \in \mathbf{O}$ that has at least one parent not in $\mathbf{O}$, the observation in $\mathbf{o}$ for O is the distinguished state o^0, then if there is a $\mathbf{O}$-active path between X and Y in $\mathcal{G}$, then $p(y^0|x^0,\mathbf{o}^0) > p(y^0|x^1,\mathbf{o}^0)$.*

Proof: Without loss of generality, assume that X is not a descendant of Y in $\mathcal{G}$. Let $UAnc(X)$ denote the set of unobserved nodes in $\mathcal{G}$ for which there is a directed path to X though unobserved nodes. In other words, $UAnc(X)$ is the set of ancestors of X if we were to remove all of the nodes in $\mathbf{O}$ from $\mathcal{G}$. We prove the theorem by induction on the size of $UAnc(X)$.

For the basis, we consider the case when $|UAnc(X)| = 0$. From Lemma 51, we can use Algorithm ONE to convert $\mathcal{G}$ into a Bayesian network containing only unobserved nodes and for which every node is BLL. Furthermore, because X has no unobserved parents in $\mathcal{G}$, we can assume by Lemma 24 that X is a root node in the resulting model $\mathcal{G}^T$. Because there is a $\mathbf{O}$-active path between X and Y in $\mathcal{G}$, there must be a $\emptyset$-active path between X and Y in $\mathcal{G}^T$. Thus the base case follows from Lemma 52.

For the induction hypothesis, we assume the theorem is true whenever $|UAnc(X)|$ is less than k, and we consider the case when $|UAnc(X)| = k$. Let Z be any element of $UAnc(X)$ for which no parent of Z is also in $UAnc(X)$; that is, Z is a root-node ancestor of X in the graph that results from removing $\mathbf{O}$ from $\mathcal{G}$. Because $Z \notin \mathbf{O}$, we know that the theorem holds if

$$\begin{array}{rcl} p(z^0|x^0,\mathbf{o}^0)p(y^0|x^0,z^0,\mathbf{o}^0) & + & p(z^1|x^0,\mathbf{o}^0)p(y^0|x^0,z^1,\mathbf{o}^0) \\ & > & \\ p(z^0|x^1,\mathbf{o}^0)p(y^0|x^1,z^0,\mathbf{o}^0) & + & p(z^1|x^1,\mathbf{o}^0)p(y^0|x^1,z^1,\mathbf{o}^0). \end{array}$$

We conclude from Proposition 29—using $\alpha_1 = p(z^0|x^0,\mathbf{o}^0)$, $\alpha_2 = p(z^0|x^1,\mathbf{o}^0)$, and $f^{ij} = p(y^0|x^i,z^j,\mathbf{o}^0)$—that the following four conditions are sufficient to establish $p(y^0|x^0,\mathbf{o}^0) \geq p(y^0|x^1,\mathbf{o}^0)$:

1. $p(z^0|x^0,\mathbf{o}^0) \geq p(z^0|x^1,\mathbf{o}^0)$ (i.e., $\alpha_1 \geq \alpha_2$)
2. $p(y^0|x^0,z^0,\mathbf{o}^0) \geq p(y^0|x^0,z^1,\mathbf{o}^0)$ (i.e., $f^{00} \geq f^{01}$)
3. $p(y^0|x^0,z^1,\mathbf{o}^0) \geq p(y^0|x^1,z^1,\mathbf{o}^0)$ (i.e., $f^{01} \geq f^{11}$)
4. $p(y^0|x^0,z^0,\mathbf{o}^0) \geq p(y^0|x^1,z^0,\mathbf{o}^0)$ (i.e., $f^{00} \geq f^{10}$)

and that either of the following two conditions is sufficient to rule out equality, and thus establish the lemma:

5. $\big(p(y^0|x^0,z^0,\mathbf{o}^0) > p(y^0|x^0,z^1,\mathbf{o}^0)\big) \wedge$ $\big(p(z^0|x^0,\mathbf{o}^0) > p(z^0|x^1,\mathbf{o}^0)\big)$ (i.e., $(\alpha_1 > \alpha_2) \wedge (f^{00} > f^{01})$)
6. $p(y^0|x^0,z^0,\mathbf{o}^0) > p(y^0|x^1,z^0,\mathbf{o}^0)$ (i.e., $f^{00} \geq f^{10}$).

We consider two cases: in $\mathcal{G}$, either X and Y are d-separated by $\mathbf{O} \cup \{Z\}$ or they are not d-separated by $\mathbf{O} \cup \{Z\}$.

Suppose X and Y are d-separated by $\mathbf{O} \cup \{Z\}$. We can immediately conclude that equality holds for both (3) and (4). Because X and Y are not d-separated by $\mathbf{O}$, we conclude both that Z and Y are d-connected given $\mathbf{O}$ and that Y and Z are d-connected given $\mathbf{O} \cup X$. From this first d-connection fact and the fact that $|UAnc(Z)| = 0$, we conclude that (1) is a strict inequality by the base case of the induction. From the second d-connection fact, and because the preconditions of the theorem are not violated by adding $X = x^0$ to the observation set, we conclude that (2) is also a strict inequality by again deferring to the base case of the induction. Thus, all inequalities (1)-(4) hold, with (1)

and (2) holding as strict inequalities. Because condition (5) is simply the conjunction of the strict versions of (2) and (1), the theorem follows.

Suppose X and Y are not d-separated by $\mathbf{O} \cup \{Z\}$. In this case, the two d-connection facts from the previous case may or may not hold. If either or both of them hold, we can show that the corresponding inequality is strict using the same argument as above. If either or both of them do not hold, we conclude that equality holds for the corresponding inequality. Thus, we know that (1) and (2) both hold, although we have no guarantees on strictness. Because all of the parents of Z are necessarily in the conditioning set, the preconditions of the theorem are not violated by adding either $z = z^0$ or $z = z^1$ to the conditioning set. Because the result of either addition reduces $|UAnc(X)|$ by one, we conclude by induction that both (3) and (4) are strict inequalities. Thus, all inequalities (1)-(4) hold. Because condition (6) is simply the strict version of (4), the theorem follows. □

Theorem 53 is closely related to existing results in the QBN literature. In particular, Theorem 4 from Druzdzel and Henrion (1993) implies that in a graph satisfying the non-strict versions of BLL and BLTSP, our Theorem 53 holds except with the conclusion that $p(y^0|x^0, \mathbf{o}^0) \geq p(y^0|x^1, \mathbf{o}^0)$.

We now prove the main results of the appendices. We re-state the corollary here, adopting our convention of using $\mathcal{G}$ to denote both the structure and the parameters of a Bayesian network.

Lemma 17 *Let $\mathcal{G}$ be a Bayesian network in which all local distributions are both BLL and BLTSP. Then the joint distribution represented by $\mathcal{G}$ is perfect with respect to the structure of $\mathcal{G}$.*

Proof: Let $p(\cdot)$ denote the joint distribution defined by $\mathcal{G}$. Because $p(\cdot)$ is defined by a Bayesian network, we know it factors according to the structure of $\mathcal{G}$, and thus we need only show that $p(\cdot)$ is faithful with respect to the structure of $\mathcal{G}$. To demonstrate that $p(\cdot)$ is faithful, we consider an arbitrary d-connection fact in $\mathcal{G}$ and prove that there is a corresponding dependence in $p(\cdot)$. Let X and Y be any pair of nodes in $\mathcal{G}$ that are d-connected by some set $\mathbf{O}$ in $\mathcal{G}$. From Theorem 53, we know that for the observation $\mathbf{O} = \mathbf{o}^0$, we have $p(y^0|x^0, \mathbf{o}^0) > p(y^0|x^1, \mathbf{o}^0)$, and thus $p(\cdot)$ is faithful. □

We now prove Lemma 5; this lemma provides a method for constructing BLL and BLTSP distributions.

Lemma 5 *Let $\mathcal{G}$ be a Bayesian network, let r_Y denote the number of states of node Y, let $\mathbf{Pa}_Y$ denote the set of parents of node Y in $\mathcal{G}$, let $NNZ(\mathbf{pa}_Y)$ denote the number of non-zero elements in the set $\mathbf{pa}_Y$, and let α_X be a constant satisfying $0 < \alpha_X < 1$. If all of the local distributions are defined as*

$$p(y|\mathbf{pa}_Y) = \begin{cases} \alpha_X^{F(\mathbf{pa}_Y)} & \text{if } y = 0 \\ \frac{1}{(r_Y - 1)} \left(1 - \alpha_X^{F(\mathbf{pa}_Y)}\right) & \text{otherwise,} \end{cases} \tag{42}$$

where

$$F(\mathbf{pa}_Y) = 2 - 2^{-NNZ(\mathbf{pa}_Y)},$$

then the distribution defined by $\mathcal{G}$ is perfect with respect to the structure of $\mathcal{G}$.

Proof: Given Lemma 17, we need only show that $p(y|\mathbf{pa}_Y)$ is BLL and BLTSP. For every variable in $\mathcal{G}$, we define the distinguished state to be state zero, and we order the states such that state zero is greatest and all non-zero states are equal. Thus, according to the definition of BLL (Definition 40) and the definition of BLTSP (Definition 41), we need to show that $p(y|\mathbf{pa}_Y)$ is binary-like, lattice with respect to $y = 0$, and totally strictly positive with respect to $y = 0$.

Due to the definition of F in Equation 42, it follows immediately from Definition 39 that $p(y|\mathbf{pa}_Y)$ is binary-like. We now show that $p(y|\mathbf{pa}_Y)$ is lattice with respect to $y = 0$. From Equation

42, this follows as long as

$$\alpha^{2-\left(\frac{1}{2}\right)^{NNZ(\mathbf{pa}_Y^1)}} > \alpha^{2-\left(\frac{1}{2}\right)^{NNZ(\mathbf{pa}_Y^2)}}$$

when $\mathbf{pa}_Y^1$ and $\mathbf{pa}_Y^2$ are identical except that $\mathbf{pa}_Y^1$ contains one extra zero in some position. Due to the fact that $\alpha < 1$, the above condition is equivalent to $NNZ(\mathbf{pa}_Y^1) < NNZ(\mathbf{pa}_Y^2)$ (simplify by taking the logarithm base α, then subtracting constants, then multiplying by -1, and then taking the logarithm base $\frac{1}{2}$; the direction of the sign above thus changes three times). Because $\mathbf{pa}_Y^1$ contains exactly one more zero than does $\mathbf{pa}_Y^2$, $NNZ(\mathbf{pa}_Y^1) = NNZ(\mathbf{pa}_Y^2) + 1$ and we conclude that $p(y|\mathbf{pa}_Y)$ is lattice with respect to $y = 0$.

Finally, we show that $p(y|\mathbf{pa}_Y)$ is totally strictly positive with respect to $y = 0$. For an arbitrary pair of parents $\{X_i, X_j\} \subseteq \mathbf{Pa}_Y$, let $\mathbf{X}_{ij}$ denote the remaining parents. That is,

$$\mathbf{Pa}_Y = \{X_i, X_j\} \cup \mathbf{X}_{ij}.$$

From Definition 27 (and the example that follows it), it suffices to show that

$$p(y = 0|x_i^0, x_j^0, \mathbf{x}_{ij})\, p(y = 0|x_i^1, x_j^1, \mathbf{x}_{ij}) > p(y = 0|x_i^0, x_j^1, \mathbf{x}_{ij})\, p(y = 0|x_i^1, x_j^0, \mathbf{x}_{ij}).$$

Letting n_{ij} denote the number of non-zero elements in $\mathbf{x}_{ij}$ and plugging in Equation 42 yields

$$\alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}+2}} \alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}}} > \alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}+1}} \alpha^{2-\left(\frac{1}{2}\right)^{n_{ij}+1}}.$$

Taking the logarithm (base α) of both sides (which reverses the sign of the inequality because $\alpha < 1$), subtracting 4 from both sides, and then dividing both sides by $-\frac{1}{2}^{n_{ij}}$ (which reverses the sign of the inequality once again) leaves

$$\left(\frac{1}{2}\right)^2 + 1 > \frac{1}{2} + \frac{1}{2},$$

which clearly holds. □

References

Bouckaert, R. R. (1995). *Bayesian Belief Networks: From Construction to Inference*. PhD thesis, Utrecht University, The Netherlands.

Chickering, D. M. (1995). A transformational characterization of Bayesian network structures. In Hanks, S. and Besnard, P., editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, QU, pages 87–98. Morgan Kaufmann.

Chickering, D. M. (1996). Learning Bayesian networks is NP-Complete. In Fisher, D. and Lenz, H., editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

Dasgupta, S. (1999). Learning polytrees. In Laskey, K. and Prade, H., editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, pages 131–141. Morgan Kaufmann.

Druzdzel, M. J. and Henrion, M. (1993). Efficient reasoning in qualitative probabilistic networks. In *Proceedings of the Eleventh Annual Conference on Artificial Intelligence (AAAI-93)*, Washington, D.C., pages 548–553.

Garey, M. and Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman.

Gavril, F. (1977). Some NP-complete problems on graphs. In *Proc. 11th Conf. on Information Sciences and Systems*, Johns Hopkins University, pages 91–95. Baltimore, MD.

Howard, R. and Matheson, J. (1981). Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, volume II, pages 721–762. Strategic Decisions Group, Menlo Park, CA.

Karlin, S. and Rinott, Y. (1980). Classes of orderings of measures and related correlation inequalities. i. multivariate totally positive distributions. *Journal of Multivariate Analysis*, 10:467–498.

Meek, C. (1997). *Graphical Models: Selecting causal and statistical models*. PhD thesis, Carnegie Mellon University.

Meek, C. (2001). Finding a path is harder than finding a tree. *Journal of Artificial Intelligence Research*, 15:383–389.

Nielsen, J. D., Kočka, T., and Peña, J. M. (2003). On local optima in learning Bayesian networks. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, Acapulco, Mexico, pages 435–442. Morgan Kaufmann.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search (second edition)*. The MIT Press, Cambridge, Massachussets.

Srebro, N. (2001). Maximum likelihood bounded tree-width Markov networks. In Breese, J. and Koller, D., editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, Seattle, WA, pages 504–511. Morgan Kaufmann.

Wellman, M. P. (1990). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303.

Journal of Machine Learning Research 5 (2004) 1331–1362 Submitted 11/02; Revised 6/03; Published 10/04

Randomized Variable Elimination

David J. Stracuzzi STRACUDJ@CS.UMASS.EDU
Paul E. Utgoff UTGOFF@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts at Amherst
140 Governors Drive
Amherst, MA 01003, USA

Editor: Haym Hirsh

Abstract

Variable selection, the process of identifying input variables that are relevant to a particular learning problem, has received much attention in the learning community. Methods that employ a learning algorithm as a part of the selection process (wrappers) have been shown to outperform methods that select variables independently from the learning algorithm (filters), but only at great computational expense. We present a randomized wrapper algorithm whose computational requirements are within a constant factor of simply learning in the presence of all input variables, provided that the number of relevant variables is small and known in advance. We then show how to remove the latter assumption, and demonstrate performance on several problems.

1. Introduction

When learning in a supervised environment, a learning algorithm is typically presented with a set of N-dimensional data points, each with its associated target output. The learning algorithm then outputs a hypothesis describing the function underlying the data. In practice, the set of N input variables is carefully selected by hand in order to improve the performance of the learning algorithm in terms of both learning speed and hypothesis accuracy.

In some cases there may be a large number of inputs available to the learning algorithm, few of which are relevant to the target function, with no opportunity for human intervention. For example, feature detectors may generate a large number of features in a pattern recognition task. A second possibility is that the learning algorithm itself may generate a large number of new concepts (or functions) in terms of existing concepts. Valiant (1984), Fahlman and Lebiere (1990), and Kivinen and Warmuth (1997) all discuss situations in which a potentially large number of features are created during the learning process. In these situations, an automatic approach to variable selection is required.

One approach to variable selection that has produced good results is the wrapper method (John et al., 1994). Here, a search is performed in the space of variable subsets, with the performance of a specific learning algorithm based on such a subset serving as an evaluation function. Using the actual generalization performance of the learning algorithm as an evaluation metric allows this approach to search for the most predictive set of input variables with respect to the learner. However, executing the learning algorithm for each selection of variables during the search ultimately renders the approach intractable in the presence of many irrelevant variables.

In spite of the cost, variable selection can play an important role in learning. Irrelevant variables can often degrade the performance of a learning algorithm, particularly when data are limited. The main computational cost associated with the wrapper method is usually that of executing the learning algorithm. The learner must produce a hypothesis for each subset of the input variables. Even greedy selection methods (Caruana and Freitag, 1994) that ignore large areas of the search space can produce a large number of candidate variable sets in the presence of many irrelevant variables.

Randomized variable elimination avoids the cost of evaluating many variable sets by taking large steps through the space of possible input sets. The number of variables eliminated in a single step depends on the number of currently selected variables. We present a cost function whose purpose is to strike a balance between the probability of failing to select successfully a set of irrelevant variables and the cost of running the learning algorithm many times. We use a form of backward elimination approach to simplify the detection of relevant variables. Removal of any relevant variable should immediately cause the learner's performance to degrade. Backward elimination also simplifies the selection process when irrelevant variables are much more common than relevant variables, as we assume here.

Analysis of our cost function shows that the cost of removing all irrelevant variables is dominated by the cost of simply learning with all N variables. The total cost is therefore within a constant factor of the cost of simply learning the target function based on all N input variables, provided that the cost of learning grows at least polynomially in N. The bound on the complexity of our algorithm is based on the complexity of the learning algorithm being used. If the given learning algorithm executes in time $O(N^2)$, then removing the $N-r$ irrelevant variables via randomized variable elimination also executes in time $O(N^2)$. This is a substantial improvement compared to the factor N or more increase experienced in removing inputs one at a time.

2. Variable Selection

The specific problem of variable selection is the following: Given a large set of input variables and a target concept or function, produce a subset of the original input variables that predict best the target concept or function when combined into a hypothesis by a learning algorithm. The term "predict best" may be defined in a variety of ways, depending on the specific application and selection algorithm. Ideally the produced subset should be as small as possible to reduce training costs and help prevent overfitting.

From a theoretical viewpoint, variable selection should not be necessary. For example, the predictive power of Bayes rule increases monotonically with the number of variables. More variables should always result in more discriminating power, and removing variables should only hurt. However, optimal applications of Bayes rule are intractable for all but the smallest problems. Many machine learning algorithms perform sub-optimal operations and do not conform to the strict conditions of Bayes rule, resulting in the potential for a performance decline in the face of unnecessary inputs. More importantly, learning algorithms usually have access to a limited number of examples. Unrelated inputs require additional capacity in the learner, but do not bring new information in exchange. Variable selection is thus a necessary aspect of inductive learning.

A variety of approaches to variable selection have been devised. Most methods can be placed into one of two categories: *filter* methods or *wrapper* methods. Filter approaches perform variable selection independently of the learning algorithm, while wrappers make learner-dependent selections. A third group of special purpose methods perform feature selection in the context of neural

networks, known as parameter pruning. These methods cannot directly perform variable selection for arbitrary learning algorithms; they are approaches to removing irrelevant inputs from learning elements.

Many variable selection algorithms (although not all) perform some form of search in the space of variable subsets as part of their operation. A forward selection algorithm begins with the empty set and searches for variables to add. A backward elimination algorithm begins with the set of all variables and searches for variables to remove. Optionally, forward algorithms may occasionally choose to remove variables, and backward algorithms may choose to add variables. This allows the search to recover from previous poor selections. The advantage of forward selection is that, in the presence of many irrelevant variables, the size of the subsets will remain relatively small, helping to speed evaluation. The advantage of backward elimination is that recognizing irrelevant variables is easier. Removing a relevant variable from an otherwise complete set should cause a decline in the evaluation, while adding a relevant variable to an incomplete set may have little immediate impact.

2.1 Filters

Filter methods use statistical measures to evaluate the quality of the variable subsets. The goal is to find a set of variables that is best with respect to the specific quality measure. Determining which variables to include may either be done via an explicit search in the space of variable subsets, or by numerically weighting the variables individually and then selecting those with the largest weight. Filter methods often have the advantage of speed. The statistical measures used to evaluate variables typically require very little computation compared to cost of running a learning algorithm many times. The disadvantage is that variables are evaluated independently, not in the context of the learning problem.

Early filtering algorithms include FOCUS (Almuallim and Dietterich, 1991) and Relief (Kira and Rendell, 1992). FOCUS searches for a smallest set of variables that can completely discriminate between target classes, while Relief ranks variables according to a distance metric. Relief selects training instances at random when computing distance values. Note that this is not related to our approach of selecting variables at random.

Decision trees have also been employed to select input variables by first inducing a tree, and then selecting only those variables tested by decision nodes (Cardie, 1993; Kubat et al., 1993). In another vein, Koller and Sahami (1996) discuss a variable selection algorithm based on cross entropy and information theory.

Methods from statistics also provide a basis for a variety of variable filtering algorithms. Correlation-based feature selection (CFS) (Hall, 1999) attempts to find a set of variables that are each highly correlated with the target function, but not with each other. The ChiMerge (Kerber, 1992) and Chi2 algorithms (Liu and Setiono, 1997) remove both irrelevant and redundant variables using a χ^2 test to merge adjacent intervals of ordinal variables.

Other methods from statistics solve problems closely related to variable selection. For example, principal component analysis (see Dunteman, 1989) is a method for transforming the observed variables into a smaller number of dimensions, as opposed to removing irrelevant or redundant variables. Projection pursuit (Friedman and Tukey, 1974) and factor analysis (Thurstone, 1931) (see Cooley and Lohnes, 1971, for a detailed presentation) are used both to reduce dimensionality and to detect structure in relationships among variables.

Discussion of filtering methods for variable selection also arises in the pattern recognition literature. For example, Devijver and Kittler (1982) discuss the use of a variety of linear and non-linear distance measures and separability measures such as entropy. They also discuss several search algorithms, such as branch and bound and plus *l*-take away *r*. Branch and bound is an optimal search technique that relies on a careful ordering of the search space to avoid an exhaustive search. Plus *l*-take away *r* is more akin to the standard forward and backward search. At each step, *l* new variables are selected for inclusion in the current set and *r* existing variables are removed.

2.2 Wrappers

Wrapper methods attempt to tailor the selection of variables to the strengths and weaknesses of specific learning algorithms by using the performance of the learner to evaluate subset quality. Each candidate variable set is evaluated by executing the learning algorithm given the selected variables and then testing the accuracy of the resulting hypotheses. This approach has the advantage of using the actual hypothesis accuracy as a measure of subset quality. The problem is that the cost of repeatedly executing the learning algorithm can quickly become prohibitive. Nevertheless, wrapper methods do tend to outperform filter methods. This is not surprising given that wrappers evaluate variables in the context of the learning problem, rather than independently.

2.2.1 ALGORITHMS

John, Kohavi, and Pfleger (1994) appear to have coined the term "wrapper" while researching the method in conjunction with a greedy search algorithm, although the technique has a longer history (Devijver and Kittler, 1982). Caruana and Freitag (1994) also experimented with greedy search methods for variable selection. They found that allowing the search to either add variables or remove them at each step of the search improved over simple forward and backward searches. Aha and Bankert (1994) use a backward elimination beam search in conjunction with the IB1 learner, but found no evidence to prefer this approach to forward selection. OBLIVION (Langley and Sage, 1994) selects variables for the nearest neighbor learning algorithm. The algorithm uses a backward elimination approach with a greedy search, terminating when the nearest neighbor accuracy begins to decline.

Subsequent work by Kohavi and John (1997) used forward and backward best-first search in the space of variable subsets. Search operators generally include adding or removing a single variable from the current set. This approach is capable of producing a minimal set of input variables, but the cost grows exponentially in the face of many irrelevant variables. Compound operators generate nodes deep in the search tree early in the search by combining the best children of a given node. However, the cost of running the best-first search ultimately remains prohibitive in the presence of many irrelevant variables.

Hoeffding races (Maron and Moore, 1994) take a different approach. All possible models (selections) are evaluated via leave-one-out cross validation. For each of the *N* evaluations, an error confidence interval is established for each model. Models whose error lower bound falls below the upper bound of the best model are discarded. The result is a set of models whose error is insignificantly different.

Several algorithms for constructing regression models are also forms of wrapper methods. For example, Least angle regression (Efron et al., 2003), which generalizes and improves upon several forward selection regression algorithms, adds variables to the model incrementally.

Genetic algorithms have been also been applied as a search mechanism for variable selection. Vafaie and De Jong (1995) describe using a genetic algorithm to perform variable selection. They used a straightforward representation in which individual chromosomes were bit-strings with each bit marking the presence or absence of a specific variable. Individuals were evaluated by training and then testing the learning algorithm. In a similar vein, SET-Gen (Cherkauer and Shavlik, 1996) used a fitness (evaluation) function that included both the accuracy of the induced model and the comprehensibility of the model. The learning model used in their experiments was a decision tree and comprehensibility was defined as a combination of tree size and number of features used. The FSS-EBNA algorithm (Inza et al., 2000) used Bayesian Networks to mate individuals in a GA-based approach to variable selection.

The relevance-in-context (RC) algorithm (Domingos, 1997) is based on the idea that some features may only be relevant in particular areas of the instance space for instance based (lazy) learners. Clusters of training examples are formed by finding examples of the same class with nearly equivalent feature vectors. The features along which the examples differ are removed and the accuracy of the entire model is determined. If the accuracy declined, the features are restored and the failed examples are removed from consideration. The algorithm continues until there are no more examples to consider. Results showed that RC outperformed other wrapper methods with respect to a 1-NN learner.

2.2.2 LEARNER SELECTIONS

Many learning algorithms already contain some (possibly indirect) form of variable selection, such as pruning in decision trees. This raises the question of whether the variable selections made by the learner should be used by the wrapper. Such an approach would almost certainly run faster than methods that rely only on the wrapper to make variable selections. The wrapper selects variables for the learner, and then executes the learner. If the resulting hypothesis is an improvement, then the wrapper further removes all variables not used in the hypothesis before continuing on with the next round of selections.

This approach assumes the learner is capable of making beneficial variable selections. If this were true, then both filter and wrapper methods would be largely irrelevant. Even the most sophisticated learning algorithms may perform poorly in the presence of highly correlated, redundant or irrelevant variables. For example, John, Kohavi, and Pfleger (1994) and Kohavi (1995) both demonstrate how C4.5 (Quinlan, 1993) can be tricked into making bad decisions *at the root*. Variables highly correlated with the target value, yet ultimately useless in terms of making beneficial data partitions, are selected near the root, leading to unnecessarily large trees. Moreover, these bad decisions cannot be corrected by pruning. Only variable selection performed outside the context of the learning algorithm can recognize these types of correlated, irrelevant variables.

2.2.3 ESTIMATING PERFORMANCE

One question that any wrapper method must consider is how to obtain a good estimate of the accuracy of the learner's hypothesis. Both the amount and quality of data available to the learner affect the testing accuracy. Kohavi and John (1997) suggest using multiple runs of five-fold cross-validation to obtain an error estimate. They determine the number of cross-validation runs by continuing until the standard deviation of the accuracy estimate is less than 1%. This has the nice property of (usually) requiring fewer runs for large data sets. However, in general, cross-validation

is an expensive procedure, requiring the learner to produce several hypotheses for each selection of variables.

2.3 Model Specific Methods

Many learning algorithms have built-in variable (or parameter) selection algorithms which are used to improve generalization. As noted above, decision tree pruning is one example of built-in variable selection. Connectionist algorithms provide several other examples, known as parameter pruning. As in the more general variable selection problem, extra weights (parameters) in a network can degrade the performance of the network on unseen test instances, and increase the cost of evaluating the learned model. Parameter pruning algorithms often suffer the same disadvantages as tree pruning. Poor choices made early in the learning process can not usually be undone.

One method for dealing with unnecessary network parameters is weight decay (Werbos, 1988). Weights are constantly pushed toward zero by a small multiplicitive factor in the update rule. Only the parameters relevant to the problem receive sufficiently large weight updates to remain significant. Methods for parameter pruning include the optimal brain damage (OBD) (LeCun et al., 1990) and optimal brain surgeon (OBS) (Hassibi and Stork, 1993) algorithms. Both rely on the second derivative to determine the importance of connection weights. Sensitivity-based pruning (Moody and Utans, 1995) evaluates the effect of removing a *network* input by replacing the input by its mean over all training points. The autoprune algorithm (Finnoff et al., 1993) defines an importance metric for weights based on the assumption that irrelevant weights will become zero. Weights with a low metric value are considered unimportant and are removed from the network.

There are also connectionist approaches that specialize in learning quickly in the presence irrelevant inputs, without actually removing them. The WINNOW algorithm (Littlestone, 1988) for Boolean functions and the exponentiated gradient algorithm (Kivinen and Warmuth, 1997) for real-valued functions are capable of learning linearly separable functions efficiently in the presence of many irrelevant variables. Exponentiated gradient algorithms, of which WINNOW is a special case, are similar to gradient descent algorithms, except that the updates are multiplicative rather than additive.

The result is a mistake bound that is linear in the number of relevant inputs, but only logarithmic in the number of irrelevant inputs. Kivinen and Warmuth also observed that the number of examples required to learn an accurate hypothesis also appears to obey these bounds. In other words, the number of training examples required by exponentiated gradient algorithms grows only logarithmicly in the number of irrelevant inputs.

Exponentiated gradient algorithms may be applied to the problem of separating the set of relevant variables from irrelevant variables by running them on the available data and examining the resulting weights. Although exponentiated gradient algorithms produce a minimum error fit of the data in non-separable problems, there is no guarantee that such a fit will rely on the variables relevant to a non-linear fit.

Many algorithms that are directly applicable in non-linear situations experience a performance decline in the presence of irrelevant input variables. Even support vector machines, which are often touted as impervious to irrelevant variables, have been shown to improve performance with feature selection (Weston et al., 2000). A more general approach to recognizing relevant variables is needed.

3. Setting

Our algorithm for randomized variable elimination (RVE) requires a set (or sequence) of N-dimensional vectors x_i with labels y_i. The learning algorithm $\mathcal{L}$ is asked to produce a hypothesis h based only on the inputs x_{ij} that have not been marked as irrelevant (alternatively, a preprocessor could remove variables marked irrelevant). We assume that the hypotheses bear some relation to the data and input values. A degenerate learner (such as one that produces the same hypothesis regardless of data or input variables) will in practice cause the selection algorithm ultimately to select zero variables. This is true of most wrapper methods. For the purposes of this article, we use generalization accuracy as the performance criteria, but this is not a requirement of the algorithm.

We make the assumption that the number r of relevant variables is at least two to avoid degenerate cases in our analysis. The number of relevant variables should be small compared to the total number of variables N. This condition is not critical to the functionality of the RVE algorithm; however the benefit of using RVE increases as the ratio of N to r increases. Importantly, we assume that the number of relevant variables is known in advance, although which variables are relevant remains hidden. Knowledge of r is a very strong assumption in practice, as such information is not typically available. We remove this assumption in Section 6, and present an algorithm for estimating r while removing variables.

4. The Cost Function

Randomized variable elimination is a wrapper method motivated by the idea that, in the presence of many irrelevant variables, the probability of successfully selecting several irrelevant variables simultaneously at random from the set of all variables is high. The algorithm computes the cost of attempting to remove k input variables out of n remaining variables given that r are relevant. A sequence of values for k is then found by minimizing the aggregate cost of removing all $N-r$ irrelevant variables. Note that n represents the number of remaining variables, while N denotes the total number of variables in the original problem.

The first step in applying the RVE algorithm is to define the cost metric for the given learning algorithm. The cost function can be based on a variety of metrics, depending on which learning algorithm is used and the constraints of the application. Ideally, a metric would indicate the amount of computational effort required for the learning algorithm to produce a hypothesis.

For example, an appropriate metric for the perceptron algorithm (Rosenblatt, 1958) might relate to the number of weight updates that must be performed, while the number of calls to the data purity criterion (e.g. information gain (Quinlan, 1986)) may be a good metric for decision tree induction algorithms. Sample complexity represents a metric that can be applied to almost any algorithm, allowing the cost function to compute the number of instances the learner must see in order to remove the irrelevant variables from the problem. We do not assume a specific metric for the definition and analysis of the cost function.

4.1 Definition

The first step of defining the cost function is to consider the probability

$$p^+(n,r,k) = \prod_{i=0}^{k-1} \left(\frac{n-r-i}{n-i} \right)$$

of successfully selecting k irrelevant variables at random and without replacement, given that there are n remaining and r relevant variables. Next we use this probability to compute the expected number of consecutive failures before a success at selecting k irrelevant variables from n remaining given that r are relevant. The expression

$$E^-(n,r,k) = \frac{1 - p^+(n,r,k)}{p^+(n,r,k)}$$

yields the expected number of consecutive trials in which at least one of the r relevant variables will be randomly selected along with irrelevant variables prior to success.

We now discuss the cost of selecting and removing k variables, given n and r. Let $M(\mathcal{L},n)$ represent an upper bound on the cost of running algorithm $\mathcal{L}$ based on n inputs. In the case of a perceptron, $M(\mathcal{L},n)$ could represent an estimated upper bound on the number of updates performed by an n-input perceptron. In some instances, such as a backpropagation neural network (Rumelhart and McClelland, 1986), providing such a bound may be troublesome. In general, the order of the worst case computational cost of the learner with respect to the number of inputs is all that is needed. The bounding function should account for any assumptions about the nature of the learning problem. For example, if learning Boolean functions requires less computational effort than learning real-valued functions, then $M(\mathcal{L},n)$ should include this difference. The general cost function described below therefore need not make any additional assumptions about the data.

In order to simplify the notation somewhat, the following discussion assumes a fixed algorithm for $\mathcal{L}$. The expected cost of successfully removing k variables from n remaining given that r are relevant is given by

$$\begin{aligned} I(n,r,k) &= E^-(n,r,k) \cdot M(\mathcal{L},n-k) + M(\mathcal{L},n-k) \\ &= M(\mathcal{L},n-k)\,(E^-(n,r,k)+1) \end{aligned}$$

for $1 \leq k \leq n-r$. The first term in the equation denotes the expected cost of failures (i.e. unsuccessful selections of k variables) while the second denotes the cost of the one success.

Given this expected cost of removing k variables, we can now define recursively the expected cost of removing all $n-r$ irrelevant variables. The goal is to minimize locally the expected cost of removing k inputs with respect to the expected remaining cost, resulting in a global minimum expected cost for removing all $n-r$ irrelevant variables. The use of a greedy minimization step relies upon the assumption that $M(\mathcal{L},n)$ is monotonic in n. This is reasonable in the context of metrics such as number of updates, number of data purity tests, and sample complexity. The cost (with respect to learning algorithm $\mathcal{L}$) of removing $n-r$ irrelevant variables is represented by

$$I_{sum}(n,r) = \min_k (I(n,r,k) + I_{sum}(n-k,r)).$$

The first part of the minimization term represents the cost of removing the first k variables while the second part represents the cost of removing the remaining $n-r-k$ irrelevant variables. Note that we define $I_{sum}(r,r) = 0$.

The optimal value $k_{opt}(n,r)$ for k given n and r can be determined in a manner similar to computing the cost of removing all $n-r$ irrelevant inputs. The value of k is computed as

$$k_{opt}(n,r) = \arg\min_k (I(n,r,k) + I_{sum}(n-k,r)).$$

4.2 Analysis

The primary benefit of this approach to variable elimination is that the combined cost (in terms of the metric $M(\mathcal{L},n)$) of learning the target function and removing the irrelevant input variables is within a constant factor of the cost of simply learning the target function based on all N inputs. This result assumes that the function $M(\mathcal{L},n)$ is at least a polynomial of degree $j > 0$. In cases where $M(\mathcal{L},n)$ is sub-polynomial, running the RVE algorithm increases the cost of removing the irrelevant inputs by a factor of $\log(n)$ over the cost of learning alone as shown below.

4.2.1 REMOVING MULTIPLE VARIABLES

We now show that the above average-case bounds on the performance of the RVE algorithm hold. The worst-case is the unlikely condition in which the algorithm always selects a relevant variable. We assume integer division here for simplicity. First let $k = \frac{n}{r}$, which allows us to remove the minimization term from the equation for $I_{sum}(n,r)$ and reduces the number of variables. This value of k is not necessarily the value selected by the above equations. However, the cost function is computed via dynamic programming, and the function $M(\mathcal{L},n)$ is assumed monotonic. Any differences between our chosen value of k and the actual value computed by the equations can only serve to decrease further the cost of the algorithm. Note also that, because k depends on the number of current variables n, k changes at each iteration of the algorithm.

The probability of success $p^+(n,r,\frac{n}{r})$ is minimized when $n = r+1$, since there is only one possible successful selection and r possible unsuccessful selections. This in turn maximizes the expected number of failures $E^-(n,r,\frac{n}{r}) = r$. The formula for $I(n,r,k)$ is now rewritten as

$$I(n,r,\frac{n}{r}) \leq (r+1)\cdot M(\mathcal{L},n-\frac{n}{r}),$$

where both $M(\mathcal{L},n-k)$ terms have been combined.

The expected cost of removing all $n-r$ irrelevant inputs may now be rewritten as a summation

$$I_{sum}(n,r) \leq \sum_{i=0}^{r\lg(n)} \left((r+1)M\left(\mathcal{L},n\left(\frac{r-1}{r}\right)^{i+1}\right)\right).$$

The second argument to the learning algorithm's cost metric M denotes the number of variables used at step i of the RVE algorithm. Notice that this number decreases geometrically toward r (recall that $n = r$ is the terminating condition for the algorithm). The logarithmic factor of the upper bound on the summation, $\frac{\lg(n)-\lg(r)}{\lg(1+1/(r-1))} \leq r\lg(n)$, follows directly from the geometric decrease in the number of variables used at each step of the algorithm. The linear factor r follows from the relationship between k and r. In general, as r increases, k decreases. Notice that as r approaches N, RVE and our cost function degrade into testing and removing variables individually.

Concluding the analysis, we observe that for functions $M(\mathcal{L},n)$ that are at least polynomial in n with degree $j > 0$, the cost incurred by the first pass of RVE ($i = 0$) will dominate the remainder of the terms. The average-case cost of running RVE in these cases is therefore bounded by $I_{sum}(N,r) \leq O(rM(\mathcal{L},N))$. An equivalent view is that the sum of a geometrically decreasing series converges to a constant. Thus, under the stated assumption that r is small compared to (and independent of) N, RVE requires only a constant factor more computation than the learner alone.

When $M(\mathcal{L},n)$ is sub-linear in n (e.g logarithmic), each pass of the algorithm contributes significantly to the total expected cost, resulting in an average-case bound of $O(r^2\log(N)M(\mathcal{L},N))$. Note

that we use average-case analysis here because in the worst case the algorithm can randomly select relevant variables indefinitely. In practice however, long streaks of bad selections are rare.

4.2.2 REMOVING VARIABLES INDIVIDUALLY

Consider now the cost of removing the $N-r$ irrelevant variables one at a time ($k=1$). Once again the probability of success is minimized and the expected number of failures is maximized at $n=r+1$. The total cost of such an approach is given by

$$I_{sum}(n,r) = \sum_{i=1}^{n-r} (r+1) \cdot M(\mathcal{L}, n-i).$$

Unlike the multiple variable removal case, the number of variables available to the learner at each step decreases only arithmetically, resulting in a linear number of steps in n. This is an important deviation from the multiple selection case, which requires only a logarithmic number of steps. The difference between the two methods becomes substantial when N is large. Concluding, the bound on the average-case cost of RVE is $I_{sum}(N,r) \leq O(NrM(\mathcal{L},N))$ when $k=1$. This is true regardless of whether the variables are selected randomly or deterministically at each step.

In principle, a comparison should be made between the upper bound of the algorithm that removes multiple variables per step and the lower bound of the algorithm that removes a single variable per step in order to show the differences clearly. However, generating a sufficiently tight lower bound requires making very strong assumptions on the form of $M(\mathcal{L},n)$. Instead, note that the two upper bounds are comparable with respect to $M(\mathcal{L},n)$ and differ only by the leading factor N.

4.3 Computing the Cost and k-Sequence

The equations for $I_{sum}(n,r)$ and $k_{opt}(n,r)$ suggest a simple $O(N^2)$ dynamic programming solution for computing both the cost and optimal k-sequence for a problem of N variables. Table 1 shows an algorithm for computing a table of cost and k values for each i with $r+1 \leq i \leq N$. The algorithm fills in the tables of values by starting with small n, and bootstrapping to find values for increasingly large n. The function $I(n,r,k)$ in Table 1 is computed as described above.

The $O(N^2)$ cost of computing the sequence of k values is of some concern. When N is large and the learning algorithm requires time only linear in N, the cost of computing the optimal k-sequence could exceed the cost of removing the irrelevant variables. In practice the cost of computing values for k is negligible for problems up to $N=1000$. For larger problems, one solution is simply to set $k=\frac{n}{r}$ as in Section 4.2.1. The analysis shows that this produces good performance and requires no computational overhead.

5. The Randomized Variable Elimination Algorithm

Randomized variable elimination conducts a backward search through the space of variable subsets, eliminating one or more variables per step. Randomization allows for selection of irrelevant variables with high probability, while selecting multiple variables allows the algorithm to move through the space without incurring the cost of evaluating the intervening points in the space. RVE conducts its search along a very narrow trajectory. The space of variable subsets is sampled sparsely, rather than broadly and uniformly. This structured yet random search allows RVE to reduce substantially the total cost of selecting relevant variables.

Given: $\mathcal{L}, N, r$

$I_{sum}[r+1..N] \leftarrow 0$
$k_{opt}[r+1..N] \leftarrow 0$

for $i \leftarrow r+1$ to N **do**
    $bestCost \leftarrow \infty$
    for $k \leftarrow 1$ to $i-r$ **do**
        $temp \leftarrow I(i,r,k) + I_{sum}[i-k]$
        if $temp < bestCost$ **then**
            $bestCost \leftarrow temp$
            $bestK \leftarrow k$
    $I_{sum}[i] \leftarrow bestCost$
    $k_{opt}[i] \leftarrow bestK$

Table 1: Algorithm for computing k and cost values.

A backward approach serves two purposes for this algorithm. First, backward elimination eases the problem of recognizing irrelevant or redundant variables. As long as a core set of relevant variables remains intact, removing other variables should not harm the performance of a learning algorithm. Indeed, the learner's performance may *increase* as irrelevant features are removed from consideration. In contrast, variables whose relevance depends on the presence of other variables may have no noticeable effect when selected in a forward manner. Thus, mistakes should be recognized immediately via backward elimination, while good selections may go unrecognized by a forward selection algorithm.

The second purpose of backward elimination is to ease the process of selecting variables. If most variables in a problem are irrelevant, then a random selection of variables is naturally likely to uncover them. Conversely, a random selection is unlikely to turn up relevant variables in a forward search. Thus, the forward search must work harder to find each relevant variable than backward search does for irrelevant variables.

5.1 Algorithm

The algorithm begins by computing the values of $k_{opt}(i,r)$ for all $r+1 \leq i \leq n$. Next it generates an initial hypothesis based on all n input variables. Then, at each step, the algorithm selects $k_{opt}(n,r)$ input variables at random for removal. The learning algorithm is trained on the remaining $n-k$ inputs, and a hypothesis h is produced. If the error $e(h')$ of hypothesis h' is less than the error $e(h)$ of the previous hypothesis h (possibly within a given tolerance), then the selected k inputs are marked as irrelevant and are all simultaneously removed from future consideration. Kohavi and John (1997) provide an in depth discussion on evaluating and comparing hypotheses based on limited data sets. If the learner was unsuccessful, meaning the new hypothesis had a larger error, then at least one of the selected variables was relevant. A new set of inputs is selected and the process repeats. The algorithm terminates when all $n-r$ irrelevant inputs have been removed. Table 2 shows the RVE algorithm.

Given: $\mathcal{L}$, n, r, *tolerance*

compute tables for $I_{sum}(i,r)$ and $k_{opt}(i,r)$
$h \leftarrow$ hypothesis produced by $\mathcal{L}$ on n inputs

while $n > r$ **do**
    $k \leftarrow k_{opt}(n,r)$
    select k variables at random and remove them
    $h' \leftarrow$ hypothesis produced by $\mathcal{L}$ on $n-k$ inputs
    if $e(h') - e(h) \leq$ *tolerance* **then**
        $n \leftarrow n-k$
        $h \leftarrow h'$
    else
        replace the selected k variables

Table 2: Randomized backward-elimination variable selection algorithm.

The structured search performed by RVE is easily distinguished from other randomized search methods. For example, genetic algorithms maintain a population of states in the search space and randomly mate the states to produce offspring with properties of both parents. The effect is an initially broad search that targets more specific areas as the search progresses. A wide variety of subsets are explored, but the cost of so much exploration can easily exceed the cost of a traditional greedy search. See Goldberg (1989) or Mitchell (1996) for detailed discussions on how genetic algorithms conduct search.

While GAs tend to drift through the search space based on the properties of individuals in the population, the LVF algorithm (Liu and Setino, 1996) samples the space of variable subsets uniformly. LVF selects both the size of each subset and the member variables at random. Although such an approach is not susceptible to "bad decisions" or local minima, the probability of finding a best or even good variable subset decreases exponentially as the number of irrelevant variables increases. Unlike RVE, LVF is a filtering method, which relies on the inconsistency rate (number of equivalent instances divided by number of total instances) in the data with respect to the selected variables.

5.2 A Simple Example

The preceding presentation of the RVE algorithm has remained strictly general, relying on no specific learning algorithm or cost metric. We consider now a specific example of how the randomized variable elimination algorithm may be applied to a linear threshold unit. The specific task examined here is to learn a Boolean function that is true when seven out of ten relevant variables are true, given a total of 100 input variables. In order to ensure that the hypotheses generated for each selection of variables has nearly minimal error, we use the thermal perceptron training algorithm (Frean, 1992). The thermal perceptron uses simulated annealing to settle weights regardless of data separability. The pocket algorithm (Gallant, 1990) is also applicable, but we found this to be slower and prone to more testing errors.

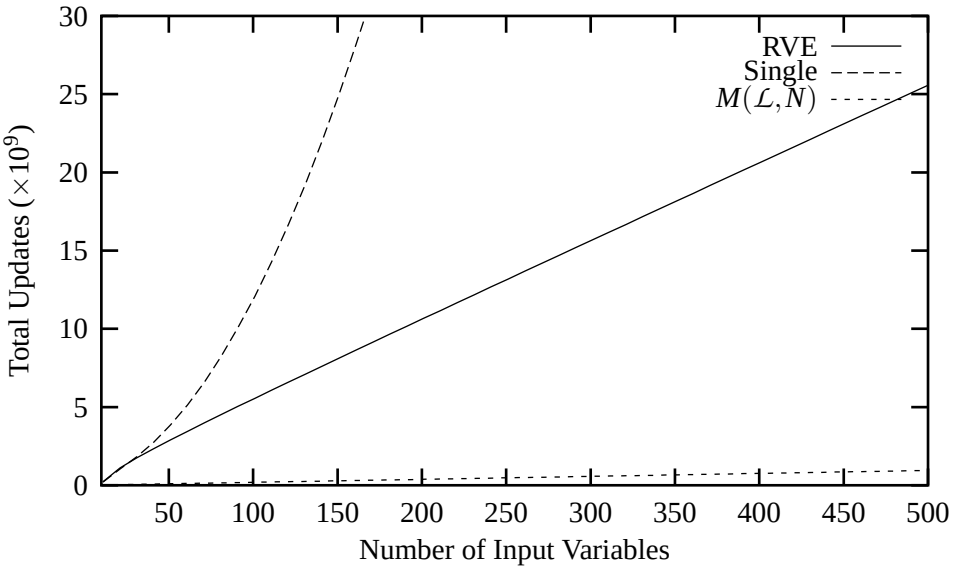


Figure 1: Plot of the expected cost of running RVE ($I_{sum}(N, r = 10)$) along with the cost of removing inputs individually, and the estimated number of updates $M(\mathcal{L},N)$.

Twenty problems were generated randomly with $N = 100$ input variables, of which 90 are irrelevant and $r = 10$ are relevant. Each of the twenty problems used a different set of ten relevant variables (selected at random) and different data sets. Two data sets, each with 1000 instances, were generated independently for each problem. One data set was used for training while the other was used to validate the error of the hypotheses generated during each round of selections. The values of the 100 input variables were all generated independently. The mean number of unique instances with respect to the ten relevant variables was 466 .

The first step in applying the RVE algorithm is to define the cost metric and the function $M(\mathcal{L},n)$ for learning on n inputs. For the perceptron, we choose the number of weight updates as the metric. The thermal perceptron anneals a temperature T that governs the magnitude of the weight updates. Here we used $T_0 = 2$ and decayed the temperature at a rate of 0.999 per training epoch until $T < 0.3$ (we observed no change in the hypotheses produced by the algorithm for $T < 0.3$). Given the temperature and decay rate, exactly 1897 training epochs are performed each time a thermal perceptron is trained. With 1000 instances in the training data, the cost of running the learning algorithm is fixed at $M(\mathcal{L},n) = 1897000(n+1)$. Given the above cost formula for an n-input perceptron, a table of values for $I_{sum}(n,r)$ and $k_{opt}(n,r)$ can be constructed.

Figure 1 plots a comparison of the computed cost of the RVE algorithm, the cost of removing variables individually, and the estimated number of updates $M(\mathcal{L},N)$ of an N-input perceptron. The calculated cost of the RVE algorithm maintains a linear growth rate with respect to N, while the cost of removing variables individually grows as N^2. This agrees with our analysis of the RVE and individual removal approaches. Relationships similar to that shown in Figure 1 arise for other values of r, although the constant factor that separates $I_{sum}(n,r)$ and $M(\mathcal{L},n)$ increases with r.

After creating the table $k_{opt}(n,r)$, the selection and removal process begins. Since the seven-of-ten learning problem is linearly separable, the tolerance for comparing the new and current hypotheses was set to near zero. A small tolerance of 0.06 (equivalent to about 15 misclassifications) is necessary since the thermal perceptron does not guarantee a minimum error hypothesis.

We also allow the current hypothesis to bias the next by not randomizing the weights (of remaining variables) after each pass of RVE. Small value weights, suggesting potential irrelevant variables, can easily transfer from one hypothesis to the next, although this is not guaranteed. Seeding the perceptron weights may increase the chance of finding a linear separator *if one exists*. If no separator exists, then seeding the weights should have minimal impact. In practice we found that the effect of seeding the weights was nullified by the pocket perceptron's use of annealing.

5.3 Example Results

The RVE algorithm was run using the twenty problems described above. Hypotheses based on ten variables were produced using an average of 5.45×10^9 weight updates, 81.1 calls to the learning algorithm, and 359.9 seconds on a 3.12 GHz Intel Xenon processor. A version of the RVE algorithm that removes variables individually (i.e. k was set permanently to 1) was also run, and produced hypotheses using 12.7×10^9 weight updates, 138.7 calls to the learner, and 644.7 seconds. These weight update values agree with the estimate produced by the cost function. Both versions of the algorithm generated hypotheses that included irrelevant and excluded relevant variables for three of the test problems. All cases in which the final selection of variables was incorrect were preceded by an initial hypothesis (based on all 100 variables) with unusually high error (error greater than 0.18 or approximately 45 misclassified instances). Thus, poor selections occured for runs in which the first hypothesis produced has high error due to annealing in the pocket perceptron.

Figure 2 plots the average number of inputs used for each variable set size (number of inputs) compared to the total number of weight updates. Each marked point on the plot denotes a size of the set of input variables given to the perceptron. The error bars indicate the standard deviation in number of updates required to reach that point. Every third point is plotted for the individual removal algorithm. Compare both the rate of drop in inputs and the number of hypotheses trained for the two RVE versions. This reflects the balance between the cost of training and unsuccessful variable selections. Removing variables individually in the presence of many irrelevant variables ignores the cost of training each hypothesis, resulting in a total cost that rises quickly early in the search process.

6. Choosing k When r Is Unknown

The assumption that the number of relevant variables r is known has played a critical role in the preceding discussion. In practice, this is a strong assumption that is not easily met. We would like an algorithm that removes irrelevant attributes efficiently without such knowledge. One approach would be simply to guess values for r and see how RVE fares. This is unsatisfying however, as a poor guess can destroy the efficiency of RVE. In general, guessing specific values for r is difficult, but placing a loose bound around r may be much easier. In some cases, the maximum value for r may be known to be much less than N, while in other cases, r can always be bounded by 1 and N.

Given some bound on the maximum r_{max} and minimum r_{min} values for r, a binary search for r can be conducted during RVE's search for relevant variables. This relies on the idea that RVE attempts to balance the cost of learning against the cost of selecting relevant variables for removal.

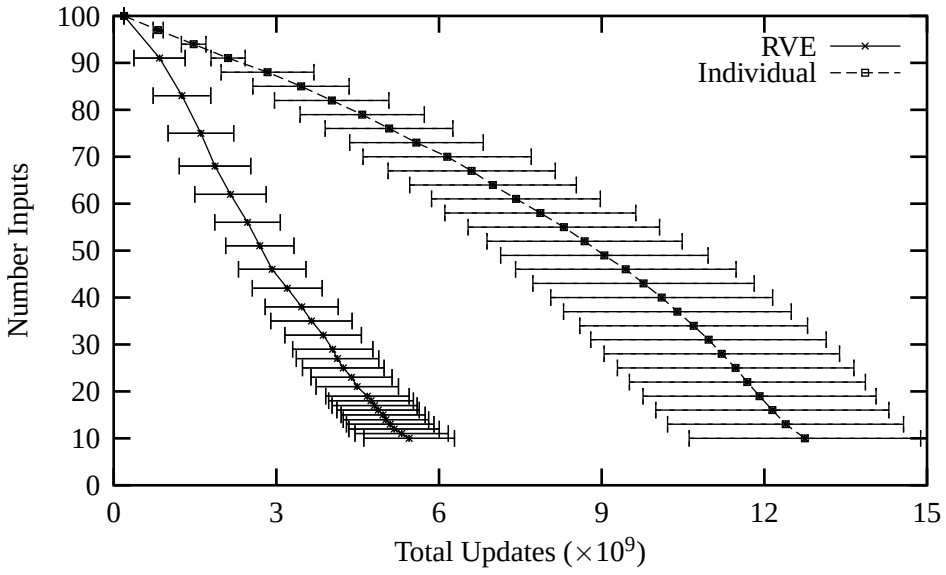


Figure 2: A comparison between the number of inputs on which the perceptrons are trained and the mean aggregate number of updates performed by the perceptrons.

At each step of RVE, a certain number of failures, $E^-(n,r,k)$, are expected. Thus, if selecting variables for removal is too easy (i.e. we are selecting too few variables at each step), then the estimate for r is too high. Similarly, if selection fails an inordinate number of times, then the estimate for r is too low.

The choice of when to adjust r is important. The selection process must be allowed to fail a certain number of times for each success, but allowing too many failures will decrease the efficiency of the algorithm. We bound the number of failures by $c_1 E^-(n,r,k)$ where $c_1 > 1$ is a constant. This allows for the failures prescribed by the cost function along with some amount of "bad luck" in the random variable selections. The number of consecutive successes is bounded similarly by $c_2(r - E^-(n,r,k))$ where $c_2 > 0$ is a constant. Since $E^-(n,r,k))$ is at most r, the value of this expression decreases as the expected number of failures increases. In practice $c_1 = 3$ and $c_2 = 0.3$ appear to work well.

6.1 A General Purpose Algorithm

Randomized variable elimination including a binary search for r (RVErS — "reverse") begins by computing tables for $k_{opt}(n,r)$ for values of r between r_{min} and r_{max}. Next an initial hypothesis is generated and the variable selection loop begins. The algorithm chooses the number of variables to remove at each step based on the current value of r. Each time the bound on the maximum number of successful selections is exceeded, r_{max} reduces to r and a new value is calculated as $r = \frac{r_{max}+r_{min}}{2}$. Similarly, when the bound on consecutive failures is exceeded, r_{min} increases to r and r is recalculated. The algorithm also checks to ensure that the current number of variables never falls below r_{min}. If this occurs, r, r_{min} and r_{max} are all set to the current number of variables. RVErS

Given: $\mathcal{L}$, c_1, c_2, n, r_{max}, r_{min}, $tolerance$

compute tables $I_{sum}(i,r)$ and $k_{opt}(i,r)$ for $r_{min} \leq r \leq r_{max}$
$r \leftarrow \frac{r_{max}+r_{min}}{2}$
$success$, $fail \leftarrow 0$
$h \leftarrow$ hypothesis produced by $\mathcal{L}$ on n inputs

repeat
    $k \leftarrow k_{opt}(n,r)$
    select k variables at random and remove them
    $h' \leftarrow$ hypothesis produced by $\mathcal{L}$ on $n-k$ inputs
    if $e(h') - e(h) \leq tolerance$ **then**
        $n \leftarrow n-k$
        $h \leftarrow h'$
        $success \leftarrow success+1$
        $fail \leftarrow 0$
    else
        replace the selected k variables
        $fail \leftarrow fail+1$
        $success \leftarrow 0$

    if $r \leq r_{min}$ **then**
        r, r_{max}, $r_{min} \leftarrow n$
    else if $fail \geq c_1 E^-(n,r,k)$ **then**
        $r_{min} \leftarrow r$
        $r \leftarrow \frac{r_{max}+r_{min}}{2}$
        $success$, $fail \leftarrow 0$
    else if $success \geq c_2(r - E^-(n,r,k))$ **then**
        $r_{max} \leftarrow r$
        $r \leftarrow \frac{r_{max}+r_{min}}{2}$
        $success$, $fail \leftarrow 0$
until $r_{min} < r_{max}$ **and** $fail \leq c_1 E^-(n,r,k)$

Table 3: Randomized variable elimination algorithm including a search for r.

terminates when r_{min} and r_{max} converge and $c_1 E^-(n,r,k)$ consecutive variable selections fail. Table 3 shows the RVE*r*S algorithm.

While RVE*r*S can produce good performance without finding the exact value of r, how well the estimated value must approximate the actual value is unclear. An important factor in determining the complexity of RVE*r*S is how quickly the algorithm reaches a good estimate for r. In the best case, the search for r will settle on a good approximation of the actual number of relevant variables immediately, and the RVE complexity bound will apply. In the worst case, the search for r will proceed slowly over values of r that are too high, causing RVE*r*S to behave like the individual removal algorithm.

Algorithm	Mean Updates	Mean Time (s)	Mean Calls	Mean Inputs
RVE (k_{opt})	5.5×10^9	359.9	81.1	10.0
$r_{max} = 20$	6.5×10^9	500.7	123.8	10.8
$r_{max} = 40$	8.0×10^9	603.8	151.3	10.2
$r_{max} = 60$	9.3×10^9	678.8	169.0	10.0
$r_{max} = 80$	10.0×10^9	694.7	172.3	10.0
$r_{max} = 100$	11.7×10^9	740.7	184.1	9.9
RVE ($k = 1$)	12.7×10^9	644.7	138.7	10.0

Table 4: Results of RVE and RVE*r*S for several values of r_{max}. Mean calls refers to the number calls made to the learning algorithm. Mean inputs refers to the number of inputs used by the final hypothesis.

With respect to the analysis presented in Section 4.2.1, note that the constants c_1 and c_2 do not impact the total cost of performing variable selection. However, a large number of adjustments to r_{min} and r_{max} do impact the total cost negatively.

6.2 An Experimental Comparison of RVE and RVE*r*S

The RVE*r*S algorithm was applied to the seven-of-ten problems using the same conditions as the experiments with RVE. Table 4 shows the results of running RVE*r*S based on five values of r_{max} and $r_{min} = 2$. The results show that for increasing values of r_{max}, the performance of RVE*r*S degrades slowly with respect to cost. The difference between RVE*r*S with $r_{max} = 100$ and RVE with $k = 1$ is significant at the 95% confidence level ($p = 0.049$), as is the difference between RVE*r*S with $r_{max} = 20$ and RVE with $k = k_{opt}$ ($p = 0.0005$). However, this slow degradation does not hold in terms of run time or number of calls to the learning algorithm. Here, only versions of RVE*r*S with $r_{max} = 20$ or 40 show an improvement over RVE with $k = 1$.

The RVE*r*S algorithm termination criteria causes the sharp increase in the number of calls to the learning algorithm. Recall that as n approaches r the probability of a failed selection increases. This means that the number of allowable selection failures grows as the algorithm nears completion. Thus, the RVE*r*S algorithm makes many calls to the learner using a small number of inputs n in an attempt to determine whether the search should be terminated. The search for r compounds the effect. If, at the end of the search, the irrelevant variables have been removed but r_{min} and r_{max} have not converged, then the algorithm must work through several failed sequences in order to terminate.

Figure 3 plots of the number of variables selected compared to the average total number of weight updates for $r_{max} = 20$, 60 and 100. The error bars represent the standard deviation in the number of updates. Notice the jump in the number of updates required for the algorithm to reach completion (represented by number of inputs equals ten) compared to the number of updates required to reach twenty remaining inputs. This pattern does not appear in the results of either version of the RVE algorithm shown in Figure 2. Traces of the RVE*r*S algorithm support the conclusion that many calls to the learner are needed to reach termination even after the correct set of variables has been found.

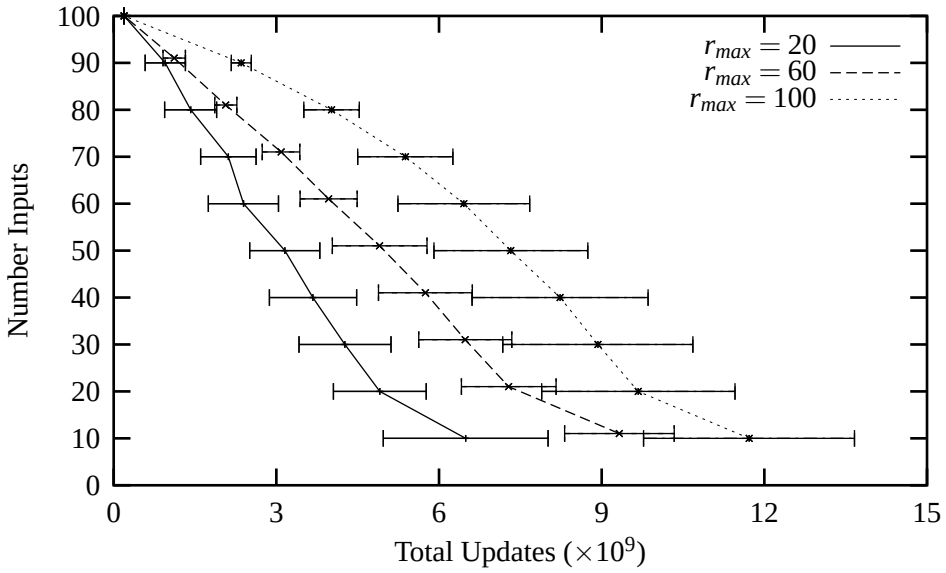


Figure 3: A comparison between the number of inputs on which the thermal perceptrons are trained and the aggregate number of updates performed using the RVE*r*S algorithm.

The increase in run times follows directly from the increasing number of calls to the learner. The thermal perceptron algorithm carries a great deal of overhead not reflected by the number of updates. Since the algorithm executes for a fixed number of epochs, the run time of any call to the learner will contribute noticeably to the run time of RVE*r*S, regardless of the number of selected variables. Contrast this behavior to that of learner whose cost is based more firmly on the number of input variables, such as naive Bayes. Thus, even though RVE*r*S always requires fewer weight updates than RVE with $k = 1$, the latter may still run faster.

This result suggests that the termination criterion of the RVE*r*S algorithm is flawed. The large number of calls to the learner at the end of the variable elimination process wastes a portion of the advantage generated earlier in the search. More importantly, the excess number of calls to the learner does not respect the very careful search trajectory computed by the cost function. Although our cost function for the learner $M(\mathcal{L}, n)$ does take the overhead of the thermal perceptron algorithm into account, there is no allowance for unnecessary calls to the learner. Future research with randomized variable elimination should therefore include a better termination criterion.

7. Experiments with RVE*r*S

We now examine the general performance properties of randomized variable elimination via experiments with several data sets. The previous experiments with perceptrons on the seven-of-ten problem focused on performance with respect to the cost metric. The following experiments are concerned primarily with minimizing run time and the number of calls to the learner while maintaining or improving accuracy. All tests were run on a 3.12 GHz Intel Xenon processor.

Data Set	Variables	Classes	Train Size	Test Size	Values of r_{max}
internet-ad	1558	2	3279	CV	1558, 750, 100
mult. feature	240	10	2000	CV	240, 150, 50
DNA	180	3	2000	1186	180, 100, 50
LED	150	10	2000	CV	150, 75, 25
opt-digits	64	10	3823	1797	64, 40, 25
soybean	35	19	683	CV	35, 25, 15
sick-euthyroid	25	2	3164	CV	25, 18, 10
monks-2-local	17	2	169	432	17, 10, 5

Table 5: Summary of data sets.

Unlike the linearly-separable perceptron experiments, the problems used here do not necessarily have solutions with zero test error. The learning algorithms may produce hypotheses with more variance in accuracy, requiring a more sophisticated evaluation function. The utility of variable selection with respect to even the most sophisticated learning algorithms is well known, see for example Kohavi and John (1997) or Weston, Mukherjee, Chapelle, Pontil, Poggio, and Vapnik (2000). The goal here is to show that our comparatively liberal elimination method sacrifices little in terms of accuracy and gains much in terms of speed.

7.1 Learning Algorithms

The RVE*r*S algorithm was applied to two learning algorithms. The first is the C4.5 release 8 algorithm (Quinlan, 1993) for decision tree induction with options to avoid pruning and early stopping. We avoid pruning and early stopping because these are forms of variable selection, and may obscure the performance of RVE*r*S. The cost metric for C4.5 is based on the number of calls to the gain-ratio data purity criterion. The cost of inducing a tree is therefore roughly quadratic in the number of variables: one call per variable, per decision node, with at most a linear number of nodes in the tree. Recall that an exact metric is not needed, only the order with respect to the number of variables must be correct.

The second learning algorithm used is naive Bayes, implemented as described by Mitchell (1997). Here, the cost metric is based on the number of operations required to build the conditional probability table, and is therefore linear in the number of inputs. In practice, these tables need not be recomputed for each new selection of variables, as the irrelevant table entries can simply be ignored. However, we recompute the tables here to illustrate the general case in which the learning algorithm must start from scratch.

7.2 Data Sets

A total of eight data sets were selected. Table 5 summarizes the data sets, and documentation is generally available from the UCI repository (Blake and Merz, 1998), except for the DNA problem, which is from StatLog (King et al., 1995). The first five problems reflect a preference for data with an abundance of variables and a large number of instances in order to demonstrate the efficiency of RVE*r*S. The last three problems are included to show how RVE*r*S performs on smaller problems, and to allow comparison with other work in variable selection. Further tests on smaller data sets

are possible, but not instructive, as randomized elimination is not intended for data sets with few variables.

Three of the data sets (DNA, opt-digits, and monks) include predetermined training and test sets. The remaining problems used ten-fold cross validation. The version of the LED problem used here was generated using code available at the repository, and includes a corruption of 10% of the class labels. Following Kohavi and John, the monks-2 data used here includes a local (one of n) encoding for each of the original six variables for a total of 17 Boolean variables. The original monks-2 problem contains no irrelevant variables, while the encoded version contains six irrelevant variables.

7.3 Methodology

For each data set and each of the two learning algorithms (C4.5 and naive Bayes), we ran four versions of the RVE*r*S algorithm. Three versions of RVE*r*S use different values of r_{max} in order to show how the choice of r_{max} affects performance. The fourth version is equivalent to RVE with $k = 1$ using a stopping criterion based on the number of consecutive failures (as in RVE*r*S). This measures the performance of removing variables individually given that the number of relevant variables is completely unknown. For comparison, we also ran forward step-wise selection, backward step-wise elimination and a hybrid filtering algorithm. The filtering algorithm simply ranked the variables by gain-ratio, executed the learner using the first 1, 2, 3, ..., N variables, and selected the best.

The learning algorithms used here provide no performance guarantees, and may produce highly variable results depending on variable selections and available data. All seven selection algorithms therefore perform five-fold cross-validation using the training data to obtain an average hypothesis accuracy generated by the learner for *each selection of variables*. The methods proposed by Kohavi and John (1997) could be used to improve error estimates for cases in which the variance in hypothesis error rates is high. Their method should provide reliable estimates for adjusting the values of r_{min} and r_{max} regardless of learning algorithm.

Preliminary experiments indicated that the RVE*r*S algorithm is more prone to becoming bogged down during the selection process than deterministic algorithms. We therefore set a small tolerance (0.002) as shown in Table 3, which allows the algorithm to keep only very good selections of variables while still preventing the selection process from stalling unnecessarily. We have not performed extensive tests to determine ideal tolerance values.

The final selections produced by the algorithms were evaluated in one of two ways. Domains for which there no specific test set is provided were evaluated via ten-fold cross-validation. The remaining domains used the provided training and test sets. In the second case, we ran each of the four RVE*r*S versions five times in order to smooth out any fluctuations due to the random nature of the algorithm.

7.4 Results

Tables 6–9 summarize the results of running the RVE*r*S algorithm on the given data sets using naive Bayes and C4.5 for learning algorithms. In the tables, *iters* denotes the number of search iterations, *evals* denotes the number of subset evaluations performed, *inputs* denotes the size of the final set of selected variables, *error* rates include the standard deviation where applicable, and *cost* represents the total cost of the search with respect to the learner's cost metric. The first row in each block shows the performance of the learner prior to variable selection, while the remaining rows show

Data Set	Learning Algorithm	Selection Algorithm	Iters	Subset Evals	Inputs	Percent Error	Time (sec)	Search Cost
internet	Bayes				1558	3.0±0.9	0.5	4.61×10^{6}
		$r_{max}=100$	137	137	37.8	3.0±1.2	165	6.13×10^{8}
		$r_{max}=750$	536	536	9.2	3.2±1.2	790	3.99×10^{9}
		$r_{max}=1558$	845	845	17.5	2.9±0.8	1406	8.26×10^{9}
		$k=1$	1658	1658	8.8	3.0±1.2	2685	1.46×10^{10}
		forward	20	30810	18.9	2.5±0.8	22417	5.32×10^{9}
		backward				NA		
		filter	1558	1558	837	3.1±0.9	2614	1.44×10^{10}
internet	C4.5				1558	3.0±0.8	48	2.04×10^{5}
		$r_{max}=100$	340	340	33.9	3.3±1.0	5386	3.30×10^{7}
		$r_{max}=750$	1233	1233	25.7	3.7±1.2	40656	2.61×10^{8}
		$r_{max}=1558$	1489	1489	20.0	3.2±1.3	78508	5.02×10^{8}
		$k=1$	1761	1761	20.6	3.3±1.0	91204	6.02×10^{8}
		forward	19	28647	17.5	3.2±1.2	18388	1.95×10^{7}
		backward				NA		
		filter	1558	1558	640	3.1±1.0	77608	3.98×10^{8}
mult-ftr	Bayes				240	34.1±4.5	0.1	4.51×10^{5}
		$r_{max}=50$	53	53	18.8	18.3±2.0	13	3.09×10^{7}
		$r_{max}=150$	84	84	19.4	17.5±4.7	27	7.28×10^{7}
		$r_{max}=240$	112	112	19.9	17.5±2.2	41	1.13×10^{8}
		$k=1$	341	341	17.2	15.7±3.0	99	2.57×10^{8}
		forward	20	4539	18.7	12.3±1.6	527	7.55×10^{8}
		backward	186	27323	55.6	13.9±1.7	12097	3.52×10^{10}
		filter	240	240	53.6	22.5±2.7	83	2.30×10^{8}
mult-ftr	C4.5				240	22.0±4.0	0.6	3.74×10^{4}
		$r_{max}=50$	306	306	22.3	22.1±2.0	241	1.13×10^{7}
		$r_{max}=150$	459	459	21.3	20.2±2.7	427	2.12×10^{7}
		$r_{max}=240$	474	474	22.0	22.1±3.5	519	2.66×10^{7}
		$k=1$	460	460	22.9	20.5±2.5	523	2.71×10^{7}
		forward	26	5960	25.3	20.4±3.5	2004	5.06×10^{7}
		backward	151	24722	90.8	20.4±3.1	51018	2.90×10^{9}
		filter	240	240	140	21.2±2.7	354	1.93×10^{7}

Table 6: Variable selection results using the naive Bayes and C4.5 learning algorithms.

the performance of the seven selection algorithms. Finally, "NA" indicates that the experiment was terminated due to excessive computational cost.

The performance of RVE*r*S on the five largest data sets is encouraging. In most cases RVE*r*S was comparable to the performance of step-wise selection with respect to generalization, while requiring substantially less computation. This effect is most clear in the mult-ftr data set, where forward selection with the C4.5 learner required nearly six CPU days to run (for ten-fold cross-validation) while the slowest RVE*r*S version required just six hours. An exception to this trend occurs with the internet-ad data using C4.5. Here, the huge cost of running C4.5 with most of the variables included overwhelms RVE*r*S's ability to eliminate variables quickly. Only the most aggressive run of the algorithm, with $r_{max}=100$, manages to bypass the problem.

Data Set	Learning Algorithm	Selection Algorithm	Iters	Subset Evals	Inputs	Percent Error	Time (sec)	Search Cost
DNA	Bayes				180	6.7	0.08	3.63×10^5
		$r_{max}=50$	359	359	24.2	4.7±0.7	52	1.52×10^8
		$r_{max}=100$	495	495	30.0	4.9±0.8	75	2.39×10^8
		$r_{max}=180$	519	519	25.6	5.0±0.5	84	2.89×10^8
		$k=1$	469	469	23.6	4.7±0.3	76	2.56×10^8
		forward	19	3249	18.0	5.8	269	2.99×10^8
		backward	34	5413	148	6.5	1399	7.16×10^9
		filter	180	180	101	5.7	32	1.33×10^8
DNA	C4.5				180	9.7	0.5	1.95×10^4
		$r_{max}=50$	356	356	17.0	8.1±1.7	198	8.42×10^6
		$r_{max}=100$	384	384	16.2	7.1±1.5	222	9.07×10^6
		$r_{max}=180$	432	432	13.8	6.5±1.2	282	1.21×10^7
		$k=1$	374	374	14.4	6.5±1.1	274	1.18×10^7
		forward	13	2262	12.0	5.9	418	2.33×10^6
		backward	110	13735	72.0	8.7	18186	8.23×10^8
		filter	180	180	17.0	7.6	163	7.10×10^6
LED	Bayes				150	30.3±3.0	0.09	2.75×10^5
		$r_{max}=25$	127	127	22.7	26.9±3.9	19	3.97×10^7
		$r_{max}=75$	293	293	17.4	26.0±3.3	50	1.09×10^8
		$r_{max}=150$	434	434	25.6	25.9±2.6	86	2.02×10^8
		$k=1$	423	423	23.7	27.0±2.1	85	2.04×10^8
		forward	14	2006	13.0	26.6±2.9	141	1.54×10^8
		backward	14	1870	138.0	30.1±2.6	667	1.95×10^9
		filter	150	150	23.7	27.1±2.1	34	8.49×10^7
LED	C4.5				150	43.9±4.5	0.5	5.48×10^4
		$r_{max}=25$	85	85	51.1	42.0±3.0	89	1.01×10^7
		$r_{max}=75$	468	468	25.8	42.5±4.5	363	3.70×10^7
		$r_{max}=150$	541	541	25.2	40.8±5.7	440	4.48×10^7
		$k=1$	510	510	32.4	42.5±2.7	439	4.63×10^7
		forward	9	1286	7.8	27.0±3.2	196	9.52×10^5
		backward	61	7218	90.9	43.5±3.5	11481	1.33×10^9
		filter	150	150	7.1	27.3±3.5	156	1.69×10^7

Table 7: Variable selection results using the naive Bayes and C4.5 learning algorithms.

The internet-ad via C4.5 experiment highlights a second point. Notice how the forward selection algorithm runs faster than all but one version of RVE*r*S. In this case, the cost and time of running C4.5 many times on a small number of variables is less than that of running C4.5 few times on many variables. However, note that a slight change in the number of iterations needed by the forward algorithm would change the time and cost of the search dramatically. This is not the case for RVE*r*S, since each iteration involves only a single evaluation instead of $O(N)$ evaluations.

The number of subset evaluations made by RVE*r*S is also important. Notice the growth in number of evaluations with respect to the total (initial) number of inputs. For aggressive versions of RVE*r*S, growth is very slow, while more conservative versions, such as $k=1$ grow approximately linearly. This suggests that the theoretical results discussed for RVE remain valid for RVE*r*S. Addi-

Data Set	Learning Algorithm	Selection Algorithm	Iters	Subset Evals	Inputs	Percent Error	Time (sec)	Search Cost
opt-digits	Bayes				64	17.4	0.08	2.59×10^5
		$r_{max}=25$	111	111	14.2	15.7±1.5	15.9	4.05×10^7
		$r_{max}=40$	157	157	13.2	14.9±0.7	22.9	5.92×10^7
		$r_{max}=64$	162	162	14.4	14.7±1.0	24.9	6.66×10^7
		$k=1$	150	150	14.0	14.2±1.1	24.7	6.76×10^7
		forward	17	952	16.0	14.1	93.8	1.91×10^8
		backward	41	1781	25.0	13.5	423.0	1.39×10^9
		filter	64	64	37.0	16.1	11.9	3.63×10^7
opt-digits	C4.5				64	43.2	0.7	2.15×10^4
		$r_{max}=25$	130	130	12.0	42.4±2.0	148	3.64×10^6
		$r_{max}=40$	158	158	10.8	42.2±0.3	181	4.42×10^6
		$r_{max}=64$	216	216	10.4	42.5±1.2	253	6.36×10^6
		$k=1$	140	140	11.4	42.1±1.1	189	5.33×10^6
		forward	16	904	15.0	41.6	645	9.64×10^6
		backward	28	1378	38.0	44.0	2842	9.55×10^7
		filter	64	64	50.0	43.6	87	2.12×10^6
soybean	Bayes				35	7.8±2.4	0.02	2.40×10^4
		$r_{max}=15$	142	142	12.6	8.9±4.2	5.9	6.47×10^6
		$r_{max}=25$	135	135	11.9	10.5±5.8	5.8	6.26×10^6
		$r_{max}=35$	132	132	11.2	9.8±5.1	5.8	6.17×10^6
		$k=1$	88	88	12.3	9.6±5.0	4.6	4.67×10^6
		forward	13	382	12.3	7.3±2.9	8.9	1.09×10^7
		backward	19	472	18.0	7.9±4.6	37.5	3.63×10^7
		filter	35	35	31.3	7.8±2.6	2.0	1.97×10^6
	C4.5				35	8.6±4.0	0.04	1.21×10^3
		$r_{max}=15$	118	118	16.3	9.5±4.6	13.5	2.78×10^5
		$r_{max}=25$	158	158	14.7	10.1±4.1	18.6	1.90×10^5
		$r_{max}=35$	139	139	16.3	9.1±3.7	17.3	3.86×10^5
		$k=1$	117	117	16.1	9.3±3.5	14.9	3.52×10^5
		forward	16	435	14.8	9.1±4.0	33.7	3.22×10^5
		backward	18	455	19.1	10.4±4.4	69.0	1.75×10^6
		filter	35	35	30.8	8.5±3.7	3.7	6.06×10^4

Table 8: Variable selection results using the naive Bayes and C4.5 learning algorithms.

tional tests using data with many hundreds or thousands of variables would be instructive, but may not be feasible with respect to the deterministic search algorithms.

RVE*r*S does not achieve the same economy of subset evaluations on the three smaller problems as on the larger problems. This is not surprising, since the ratio of relevant variables to total variables is much smaller, requiring RVE*r*S to proceed more cautiously. In these cases, the value of r_{max} has only a minor effect on performance, as RVE*r*S is unable to remove more than two or three variables in any given step.

One problem evidenced by both large and small data sets is that there appears to be no clear choice of a best value for r_{max}. Conservative versions of RVE*r*S tend to produce lower error rates,

Data Set	Learning Algorithm	Selection Algorithm	Iters	Subset Evals	Inputs	Percent Error	Time (sec)	Search Cost
euthyroid	Bayes				25	6.2±1.3	0.02	7.54×10^4
		$r_{max}=10$	30	30	2.0	4.6±1.5	2.1	2.65×10^6
		$r_{max}=18$	39	39	2.0	4.8±1.2	1.3	3.73×10^6
		$r_{max}=25$	46	46	2.3	5.1±1.4	1.6	4.32×10^6
		$k=1$	35	35	1.7	5.0±1.2	1.5	4.86×10^6
		forward	5	118	4.2	4.6±1.1	3.4	6.45×10^6
		backward	16	263	11.3	4.2±1.3	14.4	5.91×10^7
		filter	25	25	4.7	4.2±0.6	1.4	4.16×10^6
	C4.5				25	2.7±1.0	0.2	1.00×10^3
		$r_{max}=10$	49	49	2.9	2.4±0.8	21.0	2.98×10^4
		$r_{max}=18$	63	63	3.3	2.2±0.7	27.6	4.58×10^4
		$r_{max}=25$	55	55	2.7	2.5±0.8	25.3	4.92×10^4
		$k=1$	54	54	3.8	2.3±0.9	29.4	6.39×10^4
		forward	7	151	5.9	2.4±0.7	51.8	3.89×10^4
		backward	16	269	11.0	2.5±0.9	200.0	6.73×10^5
		filter	25	25	15.2	2.7±1.1	15.4	3.60×10^4
monks-2	Bayes				17	39.4	0.01	3.11×10^3
		$r_{max}=5$	25	25	2.6	36.1±3.2	0.02	1.10×10^5
		$r_{max}=10$	54	54	4.0	37.2±2.3	0.05	2.50×10^5
		$r_{max}=17$	74	74	6.0	37.4±3.1	0.08	4.93×10^5
		$k=1$	41	41	6.0	36.8±3.1	0.04	2.89×10^5
		forward	2	33	1.0	32.9	0.02	6.70×10^4
		backward	8	99	11.0	38.4	0.13	9.93×10^5
		filter	17	17	2.0	40.3	0.01	1.21×10^5
	C4.5				17	23.6	0.03	5.14×10^2
		$r_{max}=5$	36	36	8.2	16.7±10.9	0.8	2.34×10^4
		$r_{max}=10$	84	84	6.2	4.6±0.4	1.9	3.74×10^4
		$r_{max}=17$	79	79	6.4	6.5±4.7	1.8	4.63×10^4
		$k=1$	55	55	6.2	4.4±0.0	1.4	4.13×10^4
		forward	2	33	1.0	32.9	0.6	3.95×10^2
		backward	13	139	6.0	4.4	3.9	1.61×10^5
		filter	17	17	13.0	35.6	0.4	1.51×10^4

Table 9: Variable selection results using the naive Bayes and C4.5 learning algorithms.

but there are exceptions. In some cases, r_{max} has very little effect on error. However, in most cases, small values of r_{max} have a distinct positive effect on run time.

The results suggest two other somewhat surprising conclusions. One is that backward elimination does not appear to have the commonly assumed positive effect on generalization. Step-wise forward selection tends to outperform step-wise backward elimination, although randomization often reduces this effect. The second conclusion is that the hybrid filter algorithm performs well in some cases, but worse than RVE*r*S and step-wise selection in most cases. Notice also that for problems with many variables, RVE*r*S runs as fast or faster than the filter. Additional experiments along these lines would be instructive.

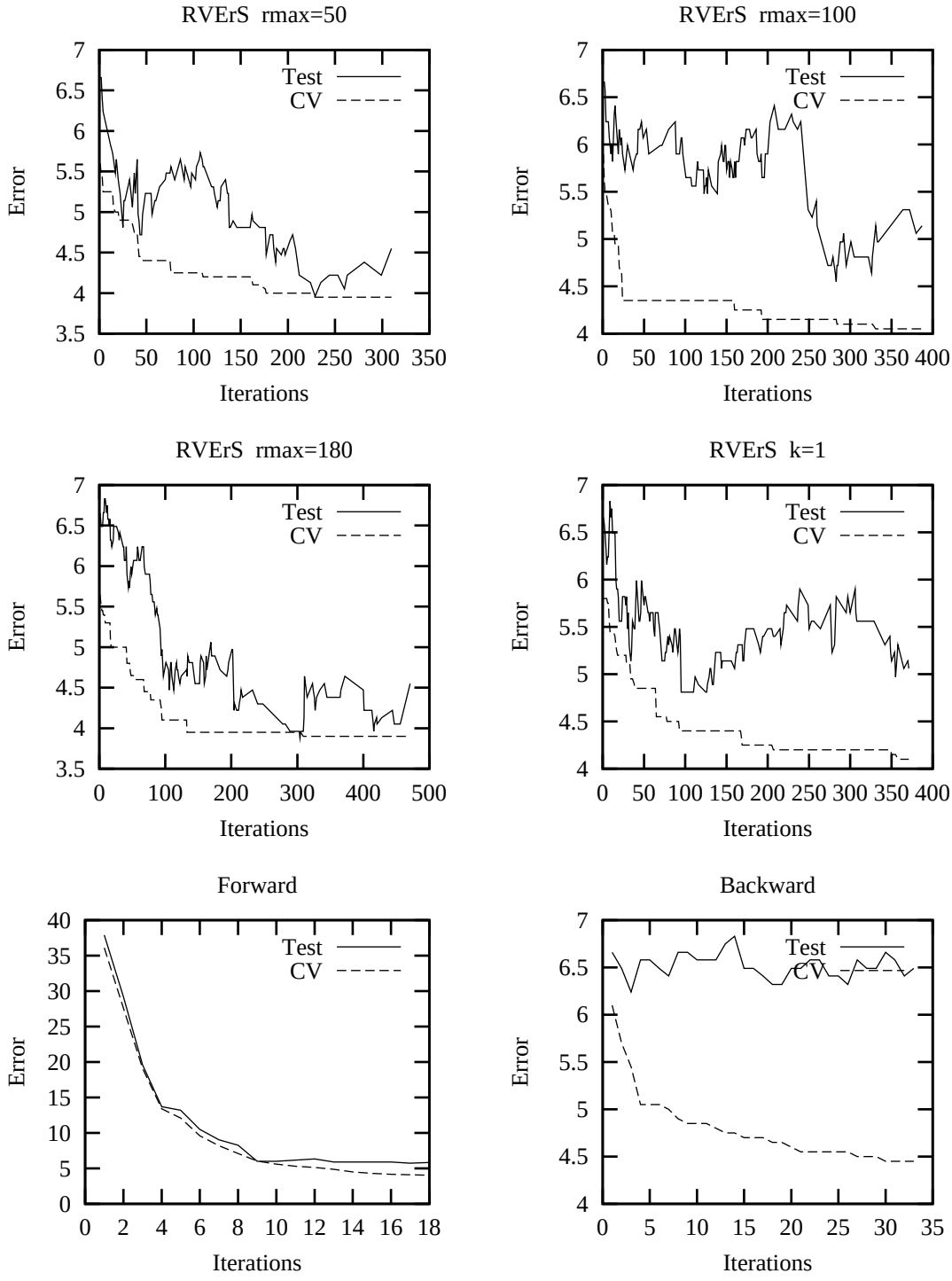


Figure 4: Naive Bayes overfitting plots for DNA data.

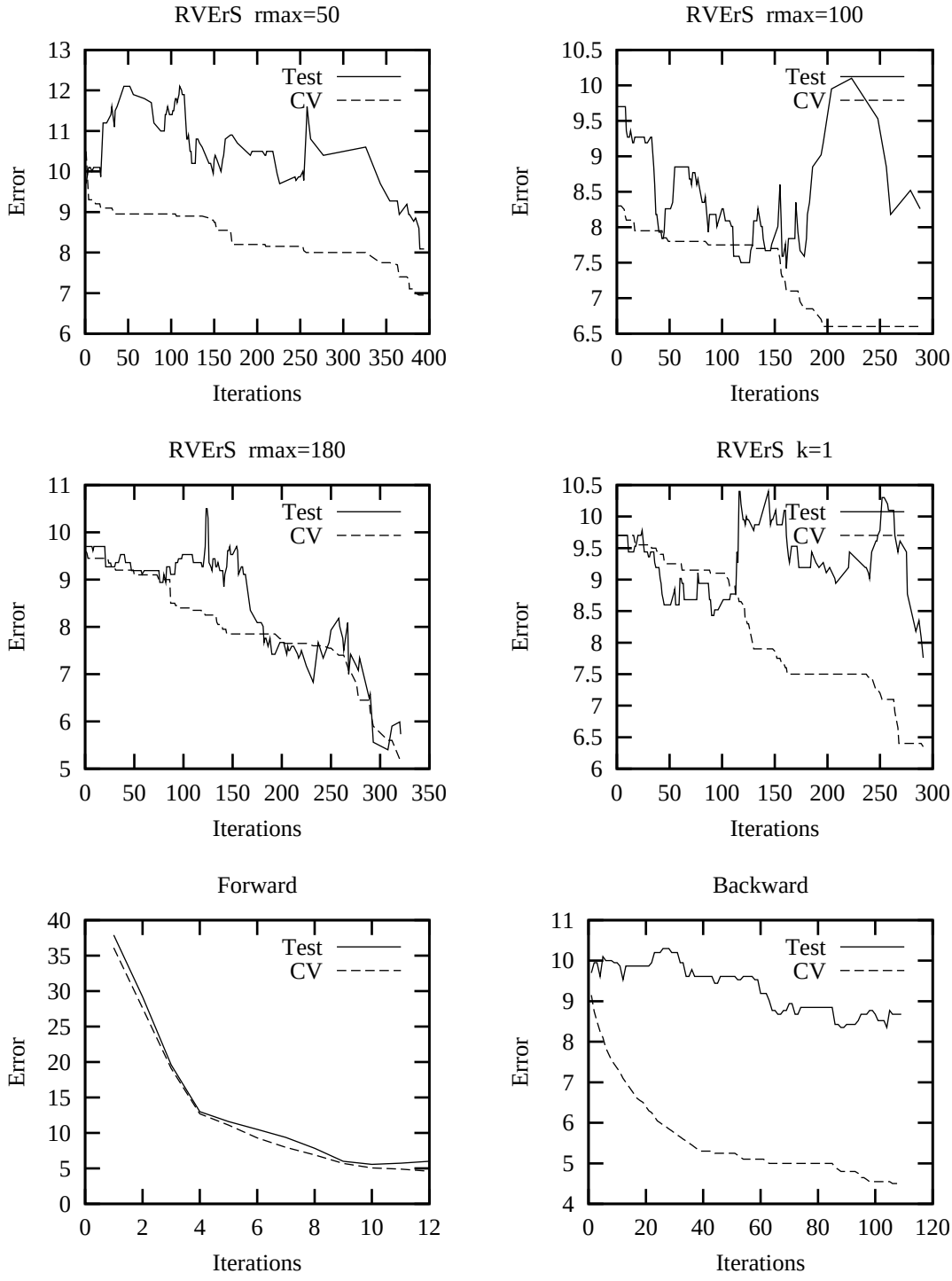


Figure 5: C4.5 overfitting plots for DNA data.

Overfitting is sometimes a problem with greedy variable selection algorithms. Figures 4 and 5 show both the test and inner (training) cross-validation error rates for the selection algorithms on naive Bayes and C4.5 respectively. Solid lines indicate test error, while dashed lines indicate the inner cross-validation error. Notice that the test error is not always minimized with the final selections produced by RVE*r*S. The graphs show that RVE*r*S does tend to overfit naive Bayes, but not C4.5 (or at least to a lesser extent). Trace data from the other data sets agree with this conclusion.

There are at least two possible explanations for overfitting by RVE*r*S. One is that the tolerance level either causes the algorithm to continue eliminating variables when it should stop, or allows elimination of relevant variables. In either case, a better adjusted tolerance level should improve performance. The monks-2 data set provides an example. In this case, if the tolerance is set to zero, RVE*r*S reliably finds variable subsets that produce low-error hypotheses with C4.5.

A second explanation is that the stopping criteria, which becomes more difficult to satisfy as the algorithm progresses, causes the elimination process to become overzealous. In this case the solution may be to augment the given stop criteria with a hold-out data set (in addition to the validation set). Here the algorithm monitors performance in addition to counting consecutive failures, returning the best selection, rather than simply the last. Combining this overfitting result with the above performance results suggests that RVE*r*S is capable of performing quite well with respect to both generalization and speed.

8. Discussion

The speed of randomized variable elimination stems from two aspects of the algorithm. One is the use of large steps in moving through the search space of variable sets. As the number of irrelevant variables grows, and the probability of selecting a relevant variable at random shrinks, RVE attempts to take larger steps toward its goal of identifying all of the *irrelevant* variables. In the face of many irrelevant variables, this is a much easier task than attempting to identify the relevant variables.

The second source of speed in RVE is the approach of removing variables immediately, instead of finding the best variable (or set) to remove. This is much less conservative than the approach taken by step-wise algorithms, and accounts for much of the benefit of RVE. In practice, the full benefit of removing multiple variables simultaneously may only be beginning to materialize in the data sets used here. However, we expect that as domains scale up, multiple selections will become increasingly important. One example of this occurs in the STL algorithm (Utgoff and Stracuzzi, 2002), which learns many concepts over a period of time. There, the number of available input variables grows as more concepts are learned by the system.

Consider briefly the cost of forward selection wrapper algorithms. Greedy step-wise search is bounded by $O(rNM(\mathcal{L},r))$ for forward selection and $O(N(N-r)M(\mathcal{L},N))$ for backward elimination, provided it does not backtrack or remove (or add) previously added (or removed) variables. The bound on the backward approach reflects both the larger number of steps required to remove the irrelevant variables and the larger number of variables used at each call to the learner. The cost of training each hypothesis is small in the forward greedy approach compared to RVE, since the number of inputs to any given hypothesis is much smaller (bounded roughly by r). However, the number of calls to the learning algorithm is polynomial in N. As the number of irrelevant variables increases, even a forward greedy approach to variable selection becomes quickly unmanageable.

The cost of a best-first search using compound operators (Kohavi and John, 1997) is somewhat harder to analyze. Their approach combines the two best operators (e.g. add variable or remove

variable) and then checks whether the result is an improvement. If so, the resulting operator is combined with the next best operator and tested, continuing until there is no improvement. Theoretically this type of search could find a solution using approximately $2r$ forward evaluations or $2(N-r)$ backward subset evaluations. However, this would require the algorithm to make the correct choice at every step. The experimental results (Kohavi and John, 1997) suggest that in practice the algorithm requires many more subset evaluations than this minimum.

Compare the above bounds on forward and backward greedy search to that of RVE given a fixed $k = 1$, which is $O(rNM(\mathcal{L}, N))$. Notice that the number of calls to the learning algorithm is the same for RVE with fixed k and a greedy *forward* search (the cost of learning is different however). The empirical results support the conclusion that the two algorithms produce similar cost, but also show that RVE with $k = 1$ requires less CPU time. The source of this additional economy is unclear, although it may be related to various overhead costs associated with the learning algorithms. RVE requires many fewer total learner executions, thereby reducing overhead.

In practice, the $k = 1$ version of RVE*r*S often makes fewer than rN calls to the learning algorithm. This follows from the very high probability of a successful selection of an irrelevant variable at each step. In cases when N is much larger than r, the algorithm with $k = 1$ makes roughly N calls to the learner as shown in Tables 6 and 7. Additional economy may also be possible when k is fixed at one. Each variable should only need to be tested once, allowing RVE*r*S to make exactly N calls to the learner. Further experiments are needed to confirm this intuition.

Although the RVE algorithm using a fixed $k = 1$ is significantly more expensive than the optimal RVE or RVE*r*S using a good guess for r_{max}, experiments and analysis show that this simple algorithm is generally faster than the deterministic forward or backward approaches, provided that there are enough irrelevant variables in the domain. As the ratio r/N decreases, and the probability of selecting an irrelevant variable at random increases, the benefit of a randomized approach improves. Thus, even when no information about the number of relevant variables is available, a randomized, backward approach to variable selection may be beneficial.

A disadvantage to randomized variable selection is that there is no clear way to recover from poor choices. Step-wise selection algorithms sometimes consider both adding and removing variables at each step, so that no variable is ever permanently selected or eliminated. A hybrid version of RVE*r*S which considers adding a single variable each time a set a variables is eliminated is possible, but this would ultimately negate much of the algorithm's computational benefit.

Step-wise selection algorithms are sometimes parallelized in order to speed the selection process. This is due in large part to the very high cost of step-wise selection. RVE mitigates this problem to a point, but there is no obvious way to parallelize a randomized selection algorithm. Parallelization could be used to improve generalization performance by allowing the algorithm to evaluate several subsets simultaneously and then choose the best.

9. Future Work

There are at least three possible directions for future work with RVE. The first is an improved method for choosing k when r is unknown. We have presented an algorithm based on a binary search, but RVE*r*S still wastes a great deal of time deciding when to terminate the search, and can quickly degenerate into a one-at-a-time removal strategy if bad decisions are made early in the search. Notice however, that this worst-case performance is still better than stepwise backward elimination, and comparable to stepwise forward selection, both popular algorithms.

A second direction for future work involves further study of the effect of testing very few of the possible successors to the current search node. Testing all possible successors is the source of the high cost of most wrapper methods. If a sparse search, such as that used by RVE, does not sacrifice much quality in general, then other fast wrapper algorithms may be possible.

A third possible direction involves biasing the random selections at each step. If a set of k variables fails to maintain evaluation performance, then at least one of the k must have been relevant to the learning problem. Thus, variables included in a failed selection may be viewed as more likely to be relevant. This "relevance likelihood" can be tracked throughout the elimination process and used to bias selections at each step.

10. Conclusion

The randomized variable elimination algorithm uses a two-step process to remove irrelevant input variables. First, a sequence of values for k, the number input variables to remove at each step, is computed such that the cost of removing all $N - r$ irrelevant variables is minimized. The algorithm then removes the irrelevant variables by randomly selecting inputs for removal according to the computed schedule. Each step is verified by generating and testing a hypothesis to ensure that the new hypothesis is at least as good as the existing hypothesis. A randomized approach to variable elimination that simultaneously removes multiple inputs produces a factor N speed-up over approaches that remove inputs individually, provided that the number r of relevant variables is known in advance.

When number of relevant variables is not known, a search for r may be conducted in parallel with the search for irrelevant variables. Although this approach wastes some of the benefits generated by the theoretical algorithm, a reasonable upper bound on the number of relevant variables still produces good performance. When even this weaker condition cannot be satisfied, a randomized approach may still outperform the conventional deterministic wrapper approaches provided that the number of relevant variables is small compared to the total number of variables. A randomized approach to variable selection is therefore applicable whenever the target domain is believed to have many irrelevant variables.

Finally, we conclude that an explicit search through the space of variable subsets is not necessary to achieve good performance from a wrapper algorithm. Randomized variable elimination provides competitive performance without incurring the high cost of expanding and evaluating all successors of a search node. As a result, randomized variable elimination scales well beyond current wrapper algorithms for variable selection.

Acknowledgments

The authors thank Bill Hesse for his advice concerning the analysis of RVE. This material is based upon work supported by the National Science Foundation under Grant No. 0097218. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

D. W. Aha and R. L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparision. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 106–112, Seattle, WA, 1994. AAAI Press.

H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, 1991. MIT Press.

C. L. Blake and C. J. Merz. Uci repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, 1998.

C. Cardie. Using decision trees to improve case-based learning. In *Machine Learning: Proceedings of the Tenth International Conference*, Amherst, MA, 1993. Morgan Kaufmann.

R. Caruana and D. Freitag. Greedy attribute selection. In *Machine Learning: Proceedings of the Eleventh International Conference*, New Brunswick, NJ, 1994. Morgan Kaufmann.

K. J. Cherkauer and J. W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

W. W. Cooley and P. R. Lohnes. *Multivariate data analysis*. Wiley, New York, 1971.

P. A. Devijver and J. Kittler. *Pattern recognition: A statistical approach*. Prentice Hall/International, 1982.

P. Domingos. Context sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997.

G. H. Dunteman. *Principal components analysis*. Sage Publications, Inc., Newbury Park CA, 1989.

B. Efron, T. Hastie, I. Johnstone, and R. Tibsharini. Least angle regression. Technical Report TR-220, Stanford University, Department of Statistics, 2003.

S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. *Advances in Neural Information Processing Systems*, 2:524–532, 1990.

W. Finnoff, F. Hergert, and H. G. Zimmermann. Improving model selection by nonconvergent methods. *Neural Networks*, 6:771–783, 1993.

M. Frean. A "thermal" perceptron learning rule. *Neural Computation*, 4(6):946–957, 1992.

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–889, 1974.

S. I. Gallant. Perceptron-based learning. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.

M. A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1999.

B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.

I. Inza, P. Larranaga, R. Etxeberria, and B. Sierra. Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence*, 123(1-2):157–184, 2000.

G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann.

R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 123–128, San Jose, CA, 1992. MIT Press.

R. King, C. Feng, and A. Shutherland. Statlog: Comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3):259–287, 1995.

K. Kira and L. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *Machine Learning: Proceedings of the Ninth International Conference*, San Mateo, CA, 1992. Morgan Kaufmann.

J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, 1997.

R. Kohavi. *Wrappers for performance enhancement and oblivious decision graphs*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 1995.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

D. Koller and M. Sahami. Toward optimal feature selection. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 284–292. Morgan Kaufmann, 1996.

M. Kubat, D. Flotzinger, and G. Pfurtscheller. Discovering patterns in Eeg signals: Comparative study of a few methods. In *Proceedings of the European Conference on Machine Learning*, pages 367–371, 1993.

P. Langley and S. Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, Seattle, WA, 1994. AAAI Press.

Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, pages 598–605, San Mateo, CA, 1990. Morgan Kaufmann.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 1988.

H. Liu and R. Setino. A probabilistic approach to feature selection: A filter solution. In *Machine Learning: Proceedings of the Fourteenth International Conference*. Morgan Kaufmann, 1996.

H. Liu and R. Setiono. Feature selection via discretization. *IEEE Transaction on Knowledge and Data Engineering*, 9(4):642–645, 1997.

O. Maron and A. W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann, 1994.

M. Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, 1996.

M. Mitchell, T. *Machine learning*. MIT Press, 1997.

J. Moody and J. Utans. Architecture selection for neural networks: Application to corporate bond rating prediction. In A. N. Refenes, editor, *Neural Networks in the Capital Markets*. John Wiley and Sons, 1995.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J. R. Quinlan. *C4.5: Machine learning programs*. Morgan Kaufmann, 1993.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and orginization in the brain. *Psychological Review*, 65:386–407, 1958.

D. E. Rumelhart and J. L. McClelland. *Parallel distributed processing*. MIT Press, Cambridge, MA, 1986. 2 volumes.

L. Thurstone. Multivariate data analysis. *Psychological Review*, 38:406–427, 1931.

P. E. Utgoff and D. J. Stracuzzi. Many-layered learning. *Neural Computation*, 14(10):2497–2529, 2002.

H. Vafaie and K. De Jong. Genetic algorithms as a tool for restructuring feature space representations. In *Proceedings of the International Conference on Tools with AI*. IEEE Computer Society Press, 1995.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

P. Werbos. Backpropagation: Past and future. In *IEEE International Conference on Neural Networks*. IEEE Press, 1988.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Advances in Neural Information Processing Systems 13*, pages 668–674. MIT Press, 2000.

Journal of Machine Learning Research 5 (2004) 1363–1390 Submitted 12/03; Revised 7/04; Published 10/04

Some Properties of Regularized Kernel Methods

Ernesto De Vito DEVITO@UNIMO.IT
Dipartimento di Matematica
Università di Modena
Modena, Italy and
INFN, Sezione di Genova
Genova, Italy

Lorenzo Rosasco ROSASCO@DISI.UNIGE.IT
Andrea Caponnetto CAPONNETTO@DISI.UNIGE.IT
DISI
Università di Genova,
Genova, Italy

Michele Piana PIANA@DIMA.UNIGE.IT
DIMA
Università di Genova,
Genova, Italy

Alessandro Verri VERRI@DISI.UNIGE.IT
DISI
Università di Genova,
Genova, Italy

Editor: Alexander J. Smola

Abstract

In regularized kernel methods, the solution of a learning problem is found by minimizing functionals consisting of the sum of a data and a complexity term. In this paper we investigate some properties of a more general form of the above functionals in which the data term corresponds to the expected risk. First, we prove a quantitative version of the representer theorem holding for both regression and classification, for both differentiable and non-differentiable loss functions, and for arbitrary offset terms. Second, we show that the case in which the offset space is non trivial corresponds to solving a standard problem of regularization in a Reproducing Kernel Hilbert Space in which the penalty term is given by a seminorm. Finally, we discuss the issues of existence and uniqueness of the solution. From the specialization of our analysis to the discrete setting it is immediate to establish a connection between the solution properties of sparsity and coefficient boundedness and some properties of the loss function. For the case of Support Vector Machines for classification, we also obtain a complete characterization of the whole method in terms of the Khun-Tucker conditions with no need to introduce the dual formulation.

Keywords: statistical learning, reproducing kernel Hilbert spaces, convex analysis, representer theorem, regularization theory

1. Introduction

The problem of learning from examples can be seen as the problem of estimating an unknown functional dependency given only a finite (possibly small) number of instances. The seminal work of Vapnik Vapnik (1988) shows that the key to effectively solve this problem is by controlling the complexity of the solution. In the context of statistical learning this leads to techniques known as *regularization networks* (Evgeniou et al., 2000) or *regularized kernel methods* (Vapnik, 1988; Cristianini and Shawe Taylor, 2000; Schölkopf and Smola, 2002). More precisely, given a training set $S = (\mathbf{x}_i, y_i)_{i=1}^{\ell}$ of ℓ pairs of examples, the estimator is defined as

$$f_S^{\lambda} \in \operatorname*{argmin}_{f \in \mathcal{H}} \left\{ \frac{1}{\ell} \sum_{i=1}^{\ell} V(y_i, f(\mathbf{x}_i)) + \lambda \| f \|_{\mathcal{H}}^2 \right\}, \tag{1}$$

where V is the loss function, $\mathcal{H}$ is the Hilbert space of the *hypotheses* and $\lambda > 0$ is the regularization parameter. As shown by Evgeniou et al. (2000) the above minimization problem can also be seen as particular instance of Tikhonov Regularization (Tikhonov and Arsenin, 1977; Mukherjee et al., 2002) for a multivariate function approximation problem which is well known to be ill-posed (Bertero et al., 1988; Evgeniou et al., 2000; Poggio and Smale, 2003).

In this paper we study the generalization of the above problem to the *continuous setting*, that is, given a probability distribution ρ defined on $X \times Y$ where X is the input space and Y is the output space, we study the properties of the estimator

$$(f^{\lambda}, g^{\lambda}) \in \operatorname*{argmin}_{(f,g) \in \mathcal{H} \times \mathcal{B}} \left\{ \int_{X \times Y} V(y, f(\mathbf{x}) + g(\mathbf{x})) d\rho(\mathbf{x}, y) + \lambda \| f \|_{\mathcal{H}}^2 \right\}, \tag{2}$$

where $\mathcal{H}$ and $\mathcal{B}$ are reproducing kernel Hilbert spaces (RKHS): $\mathcal{H}$ is the space of penalized functions and $\mathcal{B}$ is the offset space (Wahba, 1990).

Considering the continuous setting is meaningful for several reasons. First, it is useful in order to study the problem of the generalization properties of kernel methods (Steinwart, 2002). To this purpose, one associates with each function $f : X \to \mathbb{R}$ its expected risk,

$$I[f] = \int_{X \times Y} V(y, f(\mathbf{x})) d\rho(\mathbf{x}, y),$$

where ρ is the unknown probability distribution describing the relation between the input $\mathbf{x} \in X$ and the output $y \in Y$. Following Cucker and Smale (2002), for regularized kernel methods the discrepancy between the expected risk of the estimator, f_S^{λ}, and the minimum obtainable risk, $\inf_{f \in \mathcal{H}} I[f]$, can be decomposed as

$$I[f_S^{\lambda}] - \inf_{f \in \mathcal{H}} I[f] = \left(I[f_S^{\lambda}] - I[f^{\lambda}] \right) + \left(I[f^{\lambda}] - \inf_{f \in \mathcal{H}} I[f] \right),$$

where the first term represents the sample error and the second term the approximation error (Niyogi and Girosi, 1999). Clearly, insight on the form of f^{λ} can be useful to obtain better bounds on both errors. Second, considering the continuous measure ρ corresponds intuitively to finding a stable solution to the learning problem in the case of infinite number of examples and, hence, gives information about the best we can do in the hypothesis space $\mathcal{H} \times \mathcal{B}$ (Mukherjee et al., 2002). Third,

we can treat both the empirical measure and the ideal unknown probability distribution in a unified framework.

The contribution of our work is threefold. First we provide a complete characterization of the explicit form of the estimator (f^λ, g^λ) given by Eq. (2) by exploiting a convexity assumption on the loss functions. Our result can be interpreted as a quantitative version of the representer theorem holding for both regression and classification and in which explicit care is taken of the offset space $\mathcal{B}$. Then, we discuss the role of the offset space $\mathcal{B}$. The starting point of our discussion is the obvious observation that the estimator given by Problem (2) is not the *pair* (f^λ, g^λ) but the *sum* $f^\lambda + g^\lambda$. In other words the natural hypothesis space is the *sum* $\mathcal{H} + \mathcal{B}$ instead of the *product* $\mathcal{H} \times \mathcal{B}$ (which is not even a space of functions from X to $\mathbb{R}$). For arbitrary loss function we prove that Problem (2) is equivalent to a kernel method defined on $\mathcal{H} + \mathcal{B}$, which is a RKHS, with a penalty term given by a seminorm. Finally, for sake of completeness, we study the issues of the existence and uniqueness for Problem (2). When $\mathcal{B}$ is not the empty set, both issues are not trivial. In particular, for $\mathcal{B}$ equal to the set of constants, we prove existence under very reasonable conditions: for example, for classification, one needs at least two examples with different labels. About uniqueness we show that, for strictly convex loss functions, one has uniqueness if and only if the space $\mathcal{B}$ is small enough to be separated by the measure ρ: for example, in the discrete setting, this last condition means that a function $g \in \mathcal{B}$ is equal to 0 if and only if $g(\mathbf{x}_i) = 0$ for all i. For the hinge loss function, which is convex but not strictly convex, we give an *ad hoc* condition in terms of number of support vectors of the two classes.

The plan of the paper is as follows. In Section 2 we discuss our contributions with respect to previous works. In Section 3 we introduce some basic concepts of learning theory and state the assumptions we make on the loss function V and hypothesis spaces $\mathcal{H}$ and $\mathcal{B}$. In Section 4 we study the form of the solution of Problem (2). In Section 5 we discuss the theoretical meaning of the offset space $\mathcal{B}$. We discuss the problem of existence and uniqueness in Section 6. In Section 7 we apply our results to the discrete setting and focus on the case of Support Vector Machines. In the appendix we recall some notions from convex analysis in infinite dimensional spaces.

2. Putting Our Work in Context

We now briefly discuss the relation between our results and the previous works on this subject. Results about the form of the solution of kernel methods are known in the literature as *representer theorems* (if $\mathcal{B}$ is not trivial they are called *semiparametric representer theorems*).

The first result in this direction is due to Kimeldorf and Wahba (1970) for the squared loss function (see also Wahba, 1990). However, the structure of the proof holds for arbitrary loss function as shown by many authors such as Cox and O'Sullivan (1990). In the framework of statistical learning, Schölkopf et al. (2001) give a proof of the representer theorem that holds for an arbitrary loss function and for any penalty term, being it a strictly increasing function of the norm. This kind of results shows that, if the $\mathcal{H}$ is a RKHS with kernel K, the estimator f_S^λ defined by Eq. (1) can be written as

$$f_S^\lambda(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i K(\mathbf{x}, \mathbf{x}_i)\,.$$

The above result holds for arbitrary loss function and for a large class of penalty terms. However, the form of the coefficients α_i is unknown.

For the squared loss function, the form of the coefficients is well known in the context of inverse problem, see, for example, Tikhonov and Arsenin (1977), and reduces to solve a linear system of equations. For arbitrary differentiable functions, this problem was studied by Poggio and Girosi (1992); Girosi (1998); Wahba (1998) where the coefficients α_i are solution of a system of algebraic equations.

This approach cannot be applied to hinge and ε-insensitive loss function (Vapnik, 1988), since they are not differentiable: the form of the coefficients α_i is recovered only through the usual dual Lagrangian formulation of the minimization problem, see, for example, Vapnik (1988); Cristianini and Shawe Taylor (2000).

Recently, Zhang (2001) gives a quantitative representer theorem in the classification setting that holds for differentiable loss function and Steinwart (2003) extends this result for arbitrary convex loss function, without using the dual problem. In these papers the form of the coefficients α_i is given in terms of a closed equation involving the subgradient of the loss function. Moreover, they are able to extend the representer theorem to the continuous setting (a study of the solution of Tikhonov regularization in the continuous setting when the square loss is used can be found also in Cucker and Smale, 2002).

This paper, using techniques similar to those of Steinwart (2003), extends the above result in the following directions:

- our result holds both for regression and classification;
- we provide a general result that holds also when the offset term is considered. The presence of the offset space forces the coefficients α_i to satisfy a system of linear equations;
- we do not assume that input space X and the output space Y are compact. In particular, for regression we can assume $Y = \mathbb{R}$;
- we provide a simpler proof than the one of Steinwart (2003) by using known results about integral convex functionals.

A discussion of the role of the offset terms can be found in Evgeniou et al. (2000) and in Poggio et al. (2002) when the space $\mathcal{B}$ reduces to the set of constant functions. The results are close to our Theorem 6, but they are proved assuming that the unit constant is in the Mercer decomposition of the kernel and for the discrete setting, while our result holds true for offset term living in arbitrary RKHS.

The problem of the existence and uniqueness is discussed in Wahba (1998) for the discrete setting and with differentiable loss functions. For arbitrary ρ the papers by Steinwart (2002, 2003) study the existence for the classification setting with offset space reduced to the constant functions. For the hinge loss and ε-insensitive loss, the problem of uniqueness is treated in Burges and Crisp (2000, 2003). Their proof is based on the dual problem and on the Kuhn-Tucker conditions. Our results subsume the cited results as special cases, but are all obtained in the more general continuous setting. In particular our results on uniqueness of SVM solution are similar to those in Burges and Crisp (2000, 2003) but do not make use of the dual formulation.

3. Notation and Assumptions

In this section we first fix the notation and then state and comment upon the basic assumptions needed to derive the results described in the rest of the paper. We start with input and output spaces.

3.1 Input and Output Spaces

As usual, we denote with X and Y the input and output spaces respectively. We assume that X is a locally compact second countable space (this assumption is satisfied for instance if X is a closed subset of $\mathbb{R}^d$) and Y is a closed subspace of $\mathbb{R}$.

We let $Z = X \times Y$ and endow it with a probability distribution ρ defined on the Borel σ-algebra of Z. We recall that, since ρ is a bounded measure and Z is second countable, ρ is a Radon measure. In practice, ρ will be either the unknown distribution describing the relation between $\mathbf{x}$ and y or the empirical measure

$$\rho_S = \frac{1}{\ell} \sum_{i=1}^{\ell} \delta_{(\mathbf{x}_i, y_i)},$$

associated with the *training set* $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$ drawn i.i.d. with respect to ρ. We now deal with loss functions.

3.2 Loss Functions

We collect the mathematical assumptions on the loss function in the following definition and we comment on the purpose of each assumption.

Definition 1 *Given $p \in [1, +\infty[$, a function $V : Y \times \mathbb{R} \to [0, +\infty[$ such that*

1. *for all $y \in Y$ the function $V(y, \cdot)$ is convex on $\mathbb{R}$;*
2. *the function V is measurable on $Y \times \mathbb{R}$;*
3. *there are $b \in [0, +\infty[$ and $a : Y \to [0, +\infty[$ such that*

$$V(y, w) \leq a(y) + b|w|^p \qquad \forall w \in \mathbb{R},\ y \in Y \tag{3}$$

$$\int_{X \times Y} a(y) d\rho(\mathbf{x}, y) \quad < \quad +\infty, \tag{4}$$

is called a p-loss function with respect to ρ.

If the context is clear, V is simply called a loss function. The convexity hypothesis is not restrictive, being satisfied by all the loss functions commonly in use. Moreover, it is powerful from a technical point of view: it allows for the use of subgradient techniques without assuming differentiability of V and makes it possible to use convex analysis tools in the study of existence and uniqueness of functional minimizers. Finally, this requirement ensures stronger bounds for the sample error (Bartlett et al., 2002; Bartlett, 2003; Bartlett et al., 2003).

Assumption 2. is a minimal requirement for defining the expected risk and it is usually satisfied since loss functions commonly in use are continuous on Z.

Condition 3. is a technical hypothesis we need in order to use results from convex integral functional analysis. For example, it is satisfied in the following cases

1. for $p = 2$, if V is the square loss function, $V(y, w) = (y - w)^2$, and

$$\int_{X \times Y} y^2 d\rho(\mathbf{x}, y) < +\infty;$$

2. for $p = 1$, if $V(y,\cdot)$ is Lipschitz on $\mathbb{R}$ with a Lipschitz constant independent of y and

$$\int_{X\times Y} V(y,0)d\rho(\mathbf{x},y) < +\infty.$$

We now restrict our analysis to some functionals studied in statistical learning.

3.3 Learning Functionals

The *expected risk* of a measurable function $f : X \to \mathbb{R}$ is defined as

$$I[f] = \int_{X\times Y} V(y, f(\mathbf{x}))d\rho(y,\mathbf{x}),$$

and can be seen as the average error obtained by the function f, where f is a possible solution of the learning problem and the probability measure ρ is unknown.

Given a training set S, a possible way to estimate $I[f]$ is to evaluate the *empirical risk*

$$I^S_{\text{emp}}[f] = \frac{1}{\ell}\sum_{i=1}^{\ell} V(y_i, f(\mathbf{x}_i)).$$

The problem of learning is to find, given the training set S, an *estimator* f effectively predicting the label of a new point. This translates in finding a function f such that its expected risk is small with high probability.

A possible way to efficiently solve the learning problem is provided by *regularized kernel methods* which amounts to solving a problem of functional minimization as Problem (1). A generalization of Problem (1) to a continuous setting is provided by Problem (2) in which the continuous measure ρ replaces the empirical measure ρ_S in the first term. In what follows we will refer to the functionals to be minimized in both Eq. (1) and Eq. (2) as *Tikhonov functionals* and to the solutions as the *regularized solutions*.

The second term of a Tikhonov functional is a *smoothness* or a *complexity* term measuring the norm of the function f in a suitable Hilbert space $\mathcal{H}$. The minimization takes place in the *hypothesis space* $\mathcal{H}\times\mathcal{B}$. We now collect the assumptions on the hypothesis space at the basis of our analysis.

3.4 Hypothesis Space

First of all, we recall the definition of reproducing kernel Hilbert space. A RKHS $\mathcal{H}$ on X with kernel $K : X \times X \to \mathbb{R}$ is defined as the unique Hilbert space of real valued functions on X such that, for all $f \in \mathcal{H}$,

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}}\rangle_{\mathcal{H}} \quad \forall \mathbf{x} \in X, \tag{5}$$

where $K_{\mathbf{x}}$ is the function on X defined by $K_{\mathbf{x}}(\mathbf{s}) = K(\mathbf{x},\mathbf{s})$.

Given a probability meausure ρ on Z and $p \in [1,+\infty[$, we say that the kernel K is p-bounded with respect to ρ if the function K is measurable on $X\times X$ and

$$\int_{X\times Y} K(\mathbf{x},\mathbf{x})^{\frac{p}{2}} d\rho(\mathbf{x},y) < +\infty. \tag{6}$$

Clearly the above condition depends only on the marginal distribution of ρ on X and ensures that $\mathcal{H}$ is a subspace of $L^p(Z,\rho)$ with continuous inclusion (see Lemma 4 in Section 4). This fact is

essential for proving our results. In particular, the p-boundedness of the kernel is fulfilled for all $p \in [1, +\infty[$ if X is compact and the kernel is continuous or if the kernel is measurable and bounded.

We are now ready to discuss the assumptions on the hypothesis space. We fix the probability measure ρ on Z and $p \in [1, +\infty[$ such that V is p-bounded with respect to ρ. We require that the space of penalized functions $\mathcal{H}$ and the space of offset functions $\mathcal{B}$ are RKHS on X such that the corresponding kernels K and $K^{\mathcal{B}}$ are p-bounded with respect to ρ. We denote the corresponding norms by $\|\cdot\|_{\mathcal{H}}$ and $\|\cdot\|_{\mathcal{B}}$. Finally, we notice that, in general, the product space $\mathcal{H} \times \mathcal{B}$ is not a RKHS.

In learning theory usually X is compact, K is continuous and $\mathcal{B}$ is the one dimensional vector space of constant functions

$$\mathcal{B} = \{ f : X \to \mathbb{R} \,|\, f(\mathbf{x}) = b,\ b \in \mathbb{R} \} = \mathbb{R}$$

with kernel $K^{\mathcal{B}}$ simply given by $K^{\mathcal{B}}(\mathbf{x}, \mathbf{s}) = 1$. Another example of offset space, which arises in approximation problems in RKHS on a bounded interval, is the space of splines of order n, whose corresponding kernel is continuous (Wahba, 1990). In both case the p-boundedness assumption is satisfied for all p. Our framework allows to treat arbitrary (possibly infinite-dimensional) offset spaces with the possibility to incorporate jumps in the offset term.

Finally, the requirement that the hypothesis space is a RKHS is due to the fact that minimization of a convex functional in a Hilbert space is easier to treat than in an arbitrary Banach space since in the former case the subgradient of the functional is an element of the space itself. Moreover, in the proofs we use extensively the reproducing property given by Eq. (5).

4. Explicit Form of the Regularized Solution

In this section we determine the explicit form of the minimizer of the Tikhonov functional introduced in the previous section. We first state the main theorem and comment on the obtained result, then we provide the mathematical proof.

4.1 Main Theorem

Theorem 2 *Let ρ be a probability measure on $X \times Y$ where X is a locally compact second countable space and Y is a closed subset of $\mathbb{R}$. Let V be a p-loss function with respect to ρ, $p \in [1, +\infty[$. Let $\mathcal{H}$ and $\mathcal{B}$ reproducing kernel Hilbert spaces such that the corresponding kernels K and $K^{\mathcal{B}}$ are p-bounded with respect to ρ. Define $q =]1, +\infty]$ such that $\frac{1}{q} + \frac{1}{p} = 1$.*

Let $\lambda > 0$ and $(f^\lambda, g^\lambda) \in \mathcal{H} \times \mathcal{B}$, then

$$(f^\lambda, g^\lambda) \in \operatorname*{argmin}_{(f,g) \in \mathcal{H} \times \mathcal{B}} \left\{ \int_{X \times Y} V(y, f(\mathbf{x}) + g(\mathbf{x})) d\rho(\mathbf{x}, y) + \lambda \|f\|_{\mathcal{H}}^2 \right\} \tag{7}$$

if and only if there is $\alpha \in L^q(Z, \rho)$ satisfying

$$\alpha(\mathbf{x}, y) \in (\partial V)(y, f^\lambda(\mathbf{x}) + g^\lambda(\mathbf{x})) \quad (x, y) \in X \times Y \text{ a.e.} \tag{8}$$

$$f^\lambda(\mathbf{s}) = -\frac{1}{2\lambda} \int_{X \times Y} K(\mathbf{s}, \mathbf{x}) \alpha(\mathbf{x}, y) d\rho(\mathbf{x}, y) \quad \mathbf{s} \in X \tag{9}$$

$$0 = \int_{X \times Y} K^{\mathcal{B}}(\mathbf{s}, \mathbf{x}) \alpha(\mathbf{x}, y) d\rho(\mathbf{x}, y) \quad \mathbf{s} \in X. \tag{10}$$

The proof of this theorem is given in the following subsection. A few important remarks are in order.

First, the theorem gives a general quantitative version of the representer theorem. The generality is obtained by considering the continuous setting which subsumes the discrete setting if the measure ρ is the empirical measure ρ_S. In this case, the integral reduces to a finite sum and we recover the well known result that $f_S^\lambda = \sum_{i=1}^\ell \alpha_i K_{\mathbf{x}_i}$, where the $\mathbf{x}_i$ form the training set. Moreover, the solution is quantitatively characterized since the coefficients α are given by Eq. (8) involving the subgradient. For differentiable loss functions in the discrete setting, Eq. (8) reduces to

$$\alpha_i = V'(y_i, f_S^\lambda(\mathbf{x}_i) + g_S^\lambda(\mathbf{x}_i)),$$

where V' denotes the derivative with respect to the second variable (Girosi, 1998; Wahba, 1998).

Second, if $\{\psi_i\}_{i=1}^m$ is a base for $\mathcal{B}$, the offset part of the solution can be written as $g^\lambda = \sum_{i=1}^m d_i \psi_i$, where the coefficients d_i are again constrained by Eq. (8). A discussion on how to solve explicitly Eq. (8) can be found in Wahba (1998). Furthermore, the presence of $\mathcal{B}$ induces a system of linear constraints on the coefficients α_i expressed by Eq. (10) that, for $\mathcal{B} = \mathbb{R}$, reduces to the well known condition

$$\sum_{i=1}^\ell \alpha_i = 0.$$

We stress that, unlike previous works, the above equation has been derived without introducing the dual formulation.

Finally, we discuss the role of Assumption 3) in Definition 1. From the proof, it is apparent that this assumption is needed to ensure the continuity of the first term in the Tikhonov functional which in the discrete setting is trivially guaranteed. Therefore, for the discrete setting Theorem 2 holds for any convex loss function. In particular, $L^q(Z, \rho_S) = \mathbb{R}^\ell$ and the condition $\alpha \in L^q(Z, \rho_S)$ is always satisfied. Back to the continuous setting, if $V(y, \cdot)$ is Lipschitz on $\mathbb{R}$ with a Lipschitz constant independent of y and

$$\int_{X \times Y} V(y, 0) d\rho(\mathbf{x}, y) < +\infty,$$

one can choose $p = 1$, so that $q = +\infty$ and condition $\alpha \in L^\infty(Z, \rho)$ means that α is bounded. For the square loss, clearly $p = 2$, so that $q = 2$ and α is square-integrable. As shown by Steinwart (2003), for classification and compact X, one can again remove Assumption 3) of Definition 1 using the fact that a convex function is locally Lipschitz and the range of possible y is bounded.

The following corollary is the restatement of the representer theorem without offset space.

Corollary 3 *With the assumptions of Theorem* 2, *let* $f^\lambda \in \mathcal{H}$ *then*

$$f^\lambda \in \operatorname*{argmin}_{f \in \mathcal{H}} \left\{ \int_{X \times Y} V(y, f(\mathbf{x})) d\rho(\mathbf{x}, y) + \lambda \| f \|_{\mathcal{H}}^2 \right\}$$

if and only if there is $\alpha \in L^q(Z, \rho)$ *satisfying*

$$\begin{aligned} \alpha(\mathbf{x}, y) &\in (\partial V)(y, f^\lambda(\mathbf{x})) \qquad (x, y) \in X \times Y \text{ a.e.} \\ f^\lambda(\mathbf{s}) &= -\frac{1}{2\lambda} \int_{X \times Y} K(\mathbf{s}, \mathbf{x}) \alpha(\mathbf{x}, y) d\rho(\mathbf{x}, y) \quad \mathbf{s} \in X. \end{aligned}$$

4.2 Proof of the Main Theorem

Before giving the proof of the theorem we discuss the proof structure, which aside from some technicalities is very simple, and is based on two lemmas. The Tikhonov functional $I[f+g]+\lambda\|f\|^2_{\mathcal{H}}$ is a convex map on $\mathcal{H}\times\mathcal{B}$, so (f^λ,g^λ) is a minimizer of the Tikhonov functional if and only if $(0,0)$ is in its subgradient, which is a subset of $\mathcal{H}\times\mathcal{B}$. Using linearity, the computation of the subgradient of the Tikhonov functional reduces to the computation of the subgradient of $I[f+g]$ and $\|f\|^2_{\mathcal{H}}$ respectively. Since the latter functional is differentiable, the subgradient evaluation is straightforward. Some care is needed for the subgradient of the former. First, we rewrite it as an integral functional on $L^p(Z,\rho)$ and then use a fundamental result of convex analysis to interchange the integral and the subgradient.

Proof [of Theorem 2] Clearly, $\lambda\|f\|^2_{\mathcal{H}}$ is continuous and, by Lemma 4, the functional $I[f+g]$ is continuous and finite. So, from item 5 of Proposition 14, one has that

$$\partial\left(I[f+g]+\lambda\|f\|^2_{\mathcal{H}}\right)=\partial(I[f+g])+\lambda\partial(\|f\|^2_{\mathcal{H}}).$$

Now, the map

$$(f,g)\to\|f\|^2_{\mathcal{H}}$$

is differentiable with derivative $(2f,0)$ and, therefore, by item 1 of Proposition 14,

$$\partial(\|f\|^2_{\mathcal{H}})=\{(2f,0)\}. \tag{11}$$

The main difficulty is the evaluation of the subgradient of the map $I[f+g]$ given in Lemma 5. By means of this lemma we obtain that the elements of the subgradient of $I[f+g]$ at (f,g) are of the form

$$\left(\int_{X\times Y}K(\mathbf{x},\cdot)\alpha(\mathbf{x},y)d\rho(\mathbf{x},y),\int_{X\times Y}K^{\mathcal{B}}(\mathbf{x},\cdot)\alpha(\mathbf{x},y)d\rho(\mathbf{x},y)\right), \tag{12}$$

where $\alpha\in L^q(Z,\rho)$ satisfies

$$\alpha(\mathbf{x},y)\in(\partial V)(y,f(\mathbf{x})+g(\mathbf{x})) \tag{13}$$

for ρ-almost all $(x,y)\in X\times Y$. Now, by combining Eq. (11) and Eq. (12), we have that the elements of the subgradient of $I[f+g]+\lambda\|f\|^2_{\mathcal{H}}$ at point (f,g) are of the form

$$\left(\int_{X\times Y}K(\mathbf{x},\cdot)\alpha(\mathbf{x},y)d\rho(\mathbf{x},y)+2\lambda f,\int_{X\times Y}K^{\mathcal{B}}(\mathbf{x},\cdot)\alpha(\mathbf{x},y)d\rho(\mathbf{x},y)\right). \tag{14}$$

where $\alpha\in L^q(Z,\rho)$ satisfies Eq. (13).

From item 3 of Proposition 14, we have that an element $(f^\lambda,g^\lambda)\in\mathcal{H}\times\mathcal{B}$ is a minimizer of $I[f+g]+\lambda\|f\|^2_{\mathcal{H}}$ if and only if $(0,0)$ belongs to the subgradient evaluated at (f^λ,g^λ). Using Eq. (14), one has that

$$f^\lambda(\mathbf{s})=-\frac{1}{2\lambda}\int_{X\times Y}\alpha(\mathbf{x},y)K(\mathbf{x},\mathbf{s})d\rho(\mathbf{x},y)$$

$$\int_{X\times Y}\alpha(\mathbf{x},y)K^{\mathcal{B}}(\mathbf{x},\mathbf{s})d\rho(\mathbf{x},y)=0.$$

where, by means of Eq. (13), $\alpha\in L^q(Z,\rho)$ satisfies Eq. (8). This ends the proof. ■

Before computing the subgradient of the map $I[f+g]$ in Lemma 5, we need to extend the definition of expected risk on $L^p(Z,\rho)$. First of all, we let

$$I_0[u] = \int_{X\times Y} V(y,u(\mathbf{x},y))\, d\rho(\mathbf{x},y) \quad u \in L^p(Z,\rho),$$

so that $I[f+g] = I_0(\mathcal{J}(f,g))$ where $\mathcal{J} : \mathcal{H}\times\mathcal{B} \to L^p(Z,\rho)$ is the linear map

$$\mathcal{J}(f,g) = f+g,$$

(the function $f+g$ is viewed in a natural way as a function on Z).

The following lemma collects some technical facts on I_0 and $\mathcal{J}$.

Lemma 4 *With the above notations,*

1. *the functional $I_0 : L^p(Z,\rho) \to [0,+\infty[$ is well-defined and continuous;*
2. *the operator $\mathcal{J} : \mathcal{H}\times\mathcal{B} \to L^p(Z,\rho)$ is well-defined and continuous.*

Proof Since the loss function V can be regarded as function on $Z\times\mathbb{R}$, that is, $V(z,w) = V(y,w)$ where $z = (\mathbf{x},y)$, one has that $I_0[u]$ is the Nemitski functional associated with V (see Appendix), that is,

$$I_0[u] = \int_Z V(z,u(z))\, d\rho(z) \quad u \in L^p(Z,\rho).$$

We claim that $I_0[u]$ is finite. Indeed, given $u \in L^p(Z,\rho)$, by Eq. (3),

$$\int_{X\times Y} V(y,u(z))\, d\rho(\mathbf{x},y) \quad \leq \quad \int_{X\times Y} a(y) + b|u(z)|^p\, d\rho(\mathbf{x},y) < +\infty.$$

The proof that I_0 is continuous can be found in Proposition III.5.1 of Ekeland and Turnbull (1983).

In order to prove the second item, we let $f \in \mathcal{H}$. Then, by Eq. (5),

$$\begin{aligned}\int_{X\times Y} |f(\mathbf{x})|^p\, d\rho(\mathbf{x},y) &= \int_{X\times Y} |\langle f, K_{\mathbf{x}}\rangle_{\mathcal{H}}|^p\, d\rho(\mathbf{x},y)\\ &\leq \|f\|_{\mathcal{H}}^p \int_{X\times Y} K(\mathbf{x},\mathbf{x})^{\frac{p}{2}}\, d\rho(\mathbf{x},y)\\ &= C\|f\|_{\mathcal{H}}^p < +\infty.\end{aligned}$$

where $C = \int_{X\times Y} K(\mathbf{x},\mathbf{x})^{\frac{p}{2}}\, d\rho(\mathbf{x},y)$ is finite since K is p-bounded (see Eq. (6)). In particular, the function $(\mathbf{x},y) \mapsto f(\mathbf{x})$ is in $L^p(Z,\rho)$ and $\|f\|_{L^p} \leq \sqrt[p]{C}\|f\|_{\mathcal{H}}$. The same relation clearly holds for $g \in \mathcal{B}$. It follows that $\mathcal{J}$ is well defined and

$$\|f+g\|_{L^p} \leq \sqrt[p]{C}\|f\|_{\mathcal{H}} + \sqrt[p]{C'}\|g\|_{\mathcal{B}}.$$

Since J is linear, it follows that $\mathcal{J}$ is continuous. ■

Finally, the following lemma computes the subgradient of $I = I_0 \circ \mathcal{J}$.

Lemma 5 *With the above notations, let* $(f,g) \in \mathcal{H} \times \mathcal{B}$*, then* $(\phi,\psi) \in \partial(I_0 \circ \mathcal{J})(f,g)$ *if and only if there is* $\alpha \in L^q(Z,\rho)$ *such that*

$$\begin{aligned} \alpha(\mathbf{x},y) &\in (\partial V)(y, f(\mathbf{x})+g(\mathbf{x})) \quad (\mathbf{x},y) \in X \times Y \text{ a.e.} \\ \phi(\mathbf{s}) &= \int_{X\times Y} K(\mathbf{s},\mathbf{x})\alpha(\mathbf{x},y)d\rho(\mathbf{x},y) \quad \mathbf{s} \in X \\ \psi(\mathbf{s}) &= \int_{X\times Y} K^{\mathcal{B}}(\mathbf{s},\mathbf{x})\alpha(\mathbf{x},y)d\rho(\mathbf{x},y) \quad \mathbf{s} \in X. \end{aligned}$$

Proof Since I_0 is finite and continuous in $0 = \mathcal{J}(0)$, by point 6 of Proposition 14, we know that

$$\partial(I_0 \circ \mathcal{J})(f,g) = \mathcal{J}^*(\partial I_0)(\mathcal{J}(f,g)), \tag{15}$$

where $\mathcal{J}^* : L^q(Z,\rho) \to \mathcal{H} \times \mathcal{B}$ is the adjoint of $\mathcal{J}$, that is,

$$\langle \mathcal{J}^*\alpha, (f,g)\rangle_{\mathcal{H}\times\mathcal{B}} = \int_{X\times Y} \alpha(\mathbf{x},y)\mathcal{J}(f,g)(x,y)\, d\rho(x,y).$$

First of all, we compute ∂I_0. Since $I_0[0] < +\infty$, we can apply Proposition 15 so that, given $u \in L^p(Z,\rho)$, then $\alpha \in (\partial I_0)(u)$ if and only if $\alpha \in L^q(Z,\rho)$ and

$$\alpha(z) \in (\partial V)(y, u(\mathbf{x},y)),$$

for ρ-almost all $(\mathbf{x},y) \in X \times Y$.

We now compute the adjoint of $\mathcal{J}$. Let $\alpha \in L^q(Z,\rho)$ and $(\phi,\psi) = \mathcal{J}^*\alpha \in \mathcal{H} \times \mathcal{B}$. Using the reproducing property of $\mathcal{H}$ and the definition of $\mathcal{J}^*$ we can write

$$\begin{aligned} \phi(\mathbf{s}) &= \langle \phi, K_{\mathbf{s}}\rangle_{\mathcal{H}} \\ &= \langle \mathcal{J}^*\alpha, (K_{\mathbf{s}},0)\rangle_{\mathcal{H}\times\mathcal{B}} = \langle \alpha, \mathcal{J}(K_{\mathbf{s}},0)\rangle_{L^2(Z,\rho)}. \end{aligned}$$

Writing the scalar product explicitly we then find

$$\phi(\mathbf{s}) = \int_{X\times Y} K(\mathbf{s},\mathbf{x})\alpha(\mathbf{x},y)d\rho(\mathbf{x},y).$$

Reasoning in the same way we find that

$$\psi(\mathbf{s}) = \int_{X\times Y} K^{\mathcal{B}}(\mathbf{s},\mathbf{x})\alpha(\mathbf{x},y)d\rho(\mathbf{x},y).$$

Replacing the above formulas in Eq. (15), we have the thesis. ■

5. Dealing with the Offset Space $\mathcal{B}$

In this section we deal with the offset term which often appears in regularized solutions. We first motivate our analysis, then state and discuss our main result on this issue. Finally, we give the proof of the results.

5.1 Motivations

In the previous section we minimized a Tikhonov functional on the set $\mathcal{H} \times \mathcal{B}$, dealing explicitly with the possible presence of an offset term in the form of the solution. Typical examples in which offset spaces arise are Support Vector Machine algorithms (Vapnik, 1988), where the offset term is a constant accounting for the translation invariance of the separating hyperplane, and penalization methods (Wahba, 1990), where the offset space is the kernel space of the penalization operator.

However, the fact that the set $\mathcal{H} \times \mathcal{B}$ is not a RKHS (in fact, it is not even a function space) makes it cumbersome to extend of typical statistical learning results to the general setting in which the offset term is considered. For example a separate analysis, with and without the offset term, is needed for measuring the complexity of the hypothesis space or studying algorithm consistency.

In this section we show that under very weak conditions the presence of an offset term is equivalent to solving a standard regularization problem with a seminorm (Wahba, 1990).

The fact that the estimator is $f^\lambda(\mathbf{x}) + g^\lambda(\mathbf{x})$ (for regression) or $\operatorname{sgn}\big(f^\lambda(\mathbf{x}) + g^\lambda(\mathbf{x})\big)$ (for classification) suggests to replace $\mathcal{H} \times \mathcal{B}$ with the sum

$$\mathcal{S} = \mathcal{H} + \mathcal{B} = \{ f + g \,|\, f \in \mathcal{H},\ g \in \mathcal{B} \}.$$

The hypothesis space $\mathcal{S}$ is a space of functions on X and, in particular, a RKHS, the kernel being the sum of the kernels of $\mathcal{H}$ and $\mathcal{B}$. In this section we show that the minimization of a Tikhonov functional on $\mathcal{H} \times \mathcal{B}$ is essentially equivalent to the minimization of an appropriate functional on $\mathcal{S}$. This provides a rigorous derivation of the following facts.

1. The equivalent functional on $\mathcal{S}$ is also a Tikhonov functional. The penalty term is a seminorm penalizing the functions in $\mathcal{S}$ orthogonal to $\mathcal{B}$ only.

2. The estimator given by the minimization of the Tikhonov functional on $\mathcal{S}$ depends only on the kernel sum.

Moreover, since the hypothesis space $\mathcal{S}$ is a RKHS, a number of classical results of learning theory follows without further effort.

Finally, we notice that the norm of $\mathcal{B}$ (hence the kernel $K^{\mathcal{B}}$) plays no role in the functional

$$I[f+g] + \lambda \|f\|^2_{\mathcal{H}},$$

that is, all kernels, whose corresponding RKHS is $\mathcal{B}$ as a vector space, give rise to the same minimizers (f^λ, g^λ). This fact is confirmed by Eq. (18) below (see also Eq. (20)).

5.2 Main Theorem

We recall that the norm in $\mathcal{S}$ is given by

$$\|f+g\|^2_{\mathcal{S}} = \inf_{\substack{f' \in \mathcal{H}, g' \in \mathcal{B} \\ f+g = f'+g'}} \left(\|f'\|^2_{\mathcal{H}} + \|g'\|^2_{\mathcal{B}} \right) \tag{16}$$

and, with respect to this norm, $\mathcal{S}$ is a RKHS on X with kernel $K + K^{\mathcal{B}}$ (Schwartz, 1964).

We are now ready to state the following result.

Theorem 6 *Let Q be the orthogonal projection on the closed subspace of $\mathcal{S}$*

$$\mathcal{S}_0 = \{s \in \mathcal{S} \mid \langle s, g \rangle_{\mathcal{S}} = 0 \ \ \forall g \in \mathcal{B}\},$$

that is the subset of functions orthogonal to $\mathcal{B}$ w.r.t. the scalar product in $\mathcal{S}$. We have the following facts.

1. *If $(f^\lambda, g^\lambda) \in \mathcal{H} \times \mathcal{B}$ is a solution of the problem*

$$\min_{(f,g)\in\mathcal{H}\times\mathcal{B}} \{I[f+g] + \lambda \|f\|^2_{\mathcal{H}}\},$$

then $s^\lambda = f^\lambda + g^\lambda \in \mathcal{S}$ is a solution of the problem

$$\min_{s\in\mathcal{S}} \{I[s] + \lambda \|Qs\|^2_{\mathcal{S}}\}$$

and $f^\lambda = Qs^\lambda$.

2. *If $s^\lambda \in \mathcal{S}$ is a solution of the problem*

$$\min_{s\in\mathcal{S}} \{I[s] + \lambda \|Qs\|^2_{\mathcal{S}}\},$$

let $f^\lambda = Qs^\lambda$ and $g^\lambda = s^\lambda - Qs^\lambda$, then

$$I[f^\lambda + g^\lambda] + \lambda \left\| f^\lambda \right\|^2_{\mathcal{H}} = \inf_{(f,g)\in\mathcal{H}\times\mathcal{B}} \{I[f+g] + \lambda \|f\|^2_{\mathcal{H}}\}.$$

In particular, if $g^\lambda \in \mathcal{B}$, then $(f^\lambda, g^\lambda) \in \mathcal{H} \times \mathcal{B}$ is a minimizer of $I[f+g] + \lambda \|f\|^2_{\mathcal{H}}$.

Before giving the proof in the following subsection we comment on this result.

First, notice that if $\mathcal{H} \cap \mathcal{B} = \{0\}$ then $\mathcal{S} = \mathcal{H} \times \mathcal{B}$ and

$$\|f+g\|^2_{\mathcal{S}} = \|f\|^2_{\mathcal{H}} + \|g\|^2_{\mathcal{B}}.$$

In this case the theorem is trivial. However, in the arbitrary case care is needed because there are functions in $\mathcal{H}$ not orthogonal to $\mathcal{B}$. Moreover, the norm $\|\cdot\|_{\mathcal{S}}$ restricted to $\mathcal{H}$ and $\mathcal{B}$ could be different from $\|\cdot\|_{\mathcal{H}}$ and $\|\cdot\|_{\mathcal{B}}$: in particular, it could happen that $(\mathcal{B}^\perp)^\perp \neq \mathcal{B}$, where the orthogonality $^\perp$ is meant with respect to the dot product in $\mathcal{S}$. This pathology is at the root of the fact that there are cases in which the problem

$$\min_{s\in\mathcal{S}} \{I[s] + \lambda \|Qs\|^2_{\mathcal{S}}\}$$

has a solution, whereas the functional $I[f+g] + \lambda \|f\|^2_{\mathcal{H}}$ does not admit a minimizer on $\mathcal{H} \times \mathcal{B}$ (see example below). In practice, since $\mathcal{H} \cap \mathcal{B}$ in most applications is finite dimensional, this pathology does not occur and the minimization problem on $\mathcal{H} \times \mathcal{B}$ is fully equivalent to the one on $\mathcal{S}$.

Second, the advantage of using the penalty term $\|f\|^2_{\mathcal{H}}$ instead of $\|Qs\|^2_{\mathcal{S}}$ is that one can solve the minimization problem without knowing the explicit form of the projection Q. Conversely, the space $\mathcal{S}$ is the natural space to address theoretical issues.

Third, we observe that since the proof does not depend on the convexity of the loss function, the theorem holds for arbitrary (positive) loss functions. However, if V satisfies the hypotheses of Definition 1, from Theorem 2 it follows that the minimizer s^λ of $I[s]+\lambda\|Qs\|_{\mathcal{S}}^2$ is of the form

$$s^\lambda(\mathbf{s}) = \int_{X\times Y} \alpha(\mathbf{x},y)\left(K(\mathbf{x},\mathbf{s})+K^{\mathcal{B}}(\mathbf{x},\mathbf{s})\right) d\rho(\mathbf{x},y)+g^\lambda(\mathbf{s}) \tag{17}$$

$$= \int_{X\times Y} \alpha(\mathbf{x},y)K(\mathbf{x},\mathbf{s})\, d\rho(\mathbf{x},y)+g^\lambda(\mathbf{s}) \tag{18}$$

where $g^\lambda \in \overline{\mathcal{B}}$ and $\alpha \in L^q(Z,\rho)$ satisfies

$$\alpha(\mathbf{x},y) \in (\partial V)(y,s^\lambda(\mathbf{x})) \tag{19}$$

$$\int_{X\times Y} \alpha(\mathbf{x},y)K^{\mathcal{B}}(\mathbf{x},\mathbf{s}) = 0. \tag{20}$$

In particular, this implies that, given $h \in \mathcal{B}$, one can replace the kernel K with $K(\mathbf{x},\mathbf{s})+h(\mathbf{x})h(\mathbf{s})$, without changing the form of the minimizer s^λ. For example if $\mathcal{B}$ is the set of constant functions, the two kernels $K(\mathbf{x},\mathbf{s})=\mathbf{x}\cdot\mathbf{s}$ and $K(\mathbf{x},\mathbf{s})=\mathbf{x}\cdot\mathbf{s}+1$ are equivalent since both penalize the functions orthogonal to 1, that is the space of linear functions.

5.3 Proof

Before giving the proof of Theorem 6 we need to prove the following technical lemma. For this purpose we recall that $\mathcal{S}_0$ was defined as

$$\mathcal{S}_0 = \{s \in \mathcal{S} \mid \langle s,g\rangle_{\mathcal{S}} = 0 \ \ \forall g \in \mathcal{B}\},$$

and Q was the corresponding orthogonal projection from $\mathcal{S}$ onto $\mathcal{S}_0$. Moreover we let $\mathcal{H}_0$ be the closed subspace of $\mathcal{H}$ given by

$$\mathcal{H}_0 = \{f \in \mathcal{H} \mid \langle f,h\rangle_{\mathcal{H}} = 0 \ \ \forall h \in \mathcal{H}\cap\mathcal{B}\}$$

and P be the corresponding orthogonal projection from $\mathcal{H}$ onto $\mathcal{H}_0$.

In order to prove the main theorem we need the following technical lemma that characterizes the space $\mathcal{S}_0$.

Lemma 7 *Let $s = f+g \in \mathcal{S}$ with $f \in \mathcal{H}$ and $g \in \mathcal{B}$, then*

$$Qs = Pf \tag{21}$$

$$\|Qs\|_{\mathcal{S}} = \|Pf\|_{\mathcal{H}} \tag{22}$$

and there is a sequence $(f_n,g_n) \in \mathcal{H}\times\mathcal{B}$ such that

$$\lim_{n\to\infty} \|Pf - f_n\|_{\mathcal{H}} = 0 \tag{23}$$

with $f_n+g_n = s$.

Equations (21) and (22) show that $\mathcal{S}_0$ and $\mathcal{H}_0$ are the same Hilbert space and, in particular, $Qs \in \mathcal{H}$. However, in general, it could happen that $s - Qs \notin \mathcal{B}$. Equation (23) is a technical trick to overcome this pathology.

Proof [of Lemma 7] To give the proof of the lemma we need some preliminary facts. Let $\mathcal{K}$ be the closed subspace of $\mathcal{H} \times \mathcal{B}$

$$\mathcal{K} = \{(f,g) \in \mathcal{H} \times \mathcal{B} \,|\, (f,h)_{\mathcal{H}} = (g,h)_{\mathcal{B}} \; \forall h \in \mathcal{H} \cap \mathcal{B}\}.$$

It is known (Schwartz, 1964) that, given $s \in \mathcal{S}$, there is a unique $(f,g) \in \mathcal{K}$ such that $s = f + g$. Moreover for all $(f',g') \in \mathcal{H} \times \mathcal{B}$,

$$\left\langle s, f' + g' \right\rangle_{\mathcal{S}} = \left\langle f, f' \right\rangle_{\mathcal{H}} + \left\langle g, g' \right\rangle_{\mathcal{B}}. \tag{24}$$

From Eq. (16) one has that

$$\|f\|_{\mathcal{S}} \leq \|f\|_{\mathcal{H}} \quad f \in \mathcal{H} \tag{25}$$

First of all we claim that $\mathcal{H}_0 \subset \mathcal{S}_0$. Clearly, if $f \in \mathcal{H}_0$, then $(f,0) \in \mathcal{K}$ and, by Eq. (24), for all $g' \in \mathcal{B}$,

$$\left\langle f + 0, 0 + g' \right\rangle_{\mathcal{S}} = \langle f, 0 \rangle_{\mathcal{H}} + \left\langle 0, g' \right\rangle_{\mathcal{B}} = 0,$$

that is $f \in \mathcal{S}_0$. This shows the claim. Moreover,

$$\|f\|_{\mathcal{S}}^2 = \langle f + 0, f + 0 \rangle_{\mathcal{S}} = \langle f, f \rangle_{\mathcal{H}} = \|f\|_{\mathcal{H}}^2. \tag{26}$$

Let $s = f + g$ with $f \in \mathcal{H}$ and $g \in \mathcal{B}$. Clearly, $f = Pf + h$ where $h \in \mathcal{H}_0^{\perp} = ((\mathcal{H} \cap \mathcal{B})^{\perp})^{\perp} = \mathcal{H} \bar{\cap} \mathcal{B}$ (here $^{\perp}$ denotes the orthogonal complement with respect to the scalar product of $\mathcal{H}$). It follows that there is a sequence $h_n \in \mathcal{H} \cap \mathcal{B}$ such that

$$\lim_{n \to \infty} \|h - h_n\|_{\mathcal{H}} = 0. \tag{27}$$

Since, by Eq. (25), $\|h - h_n\|_{\mathcal{S}} \leq \|h - h_n\|_{\mathcal{H}}$ and Q is continuous, it follows that $Qh = \lim_{n \to \infty} Qh_n = 0$, since $Qh_n = 0$. The statements of the theorem easily follow from the above facts. Indeed

$$Qs = Q(Pf + h + g) = QPf = Pf,$$

since $Pf \in \mathcal{H}_0 \subset \mathcal{S}_0$, and Equation (21) is proved. Equation (22) follows from Eq. (26). Finally let now $f_n = Pf + h - h_n$ and $g_n = g + h_n$. Clearly, $f_n + g_n = f + g = s$, $f_n \in \mathcal{H}$ and $g_n \in \mathcal{B}$ and moreover Eq. (23) follows from Eq. (27). ■

We are now ready to prove the main theorem of this section.

Proof [Theorem 6] First of all we note the following facts. Let $f \in \mathcal{H}$, $g \in \mathcal{B}$ and $s = f + g \in \mathcal{S}$. By Eq. (22)

$$I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 = I[f + g] + \lambda \|Pf\|_{\mathcal{H}}^2 \tag{28}$$

Let $(f_n, g_n) \in \mathcal{H} \times \mathcal{B}$ as in Lemma 7, then

$$I[f + g] + \lambda \|Pf\|_{\mathcal{H}}^2 = \lim_n \left(I[f_n + g_n] + \lambda \|f_n\|_{\mathcal{H}}^2 \right).$$

From the above equalities it follows that

$$I[s]+\lambda\|Qs\|_{\mathcal{S}}^2=\lim_n\left(I[f_n+g_n]+\lambda\|f_n\|_{\mathcal{H}}^2\right). \tag{29}$$

We can now prove the first part of the theorem. Assume that $(f^\lambda,g^\lambda)\in\mathcal{H}\times\mathcal{B}$ is a minimizer of $I[f+g]+\lambda\|f\|_{\mathcal{H}}^2$ and let $s^\lambda=f^\lambda+g^\lambda$. From Eq. (29) and the definition of minimizer, one has that, for all $s\in\mathcal{S}$,

$$I[s]+\lambda\|Qs\|_{\mathcal{S}}^2\geq I[f^\lambda+g^\lambda]+\lambda\left\|f^\lambda\right\|_{\mathcal{H}}^2. \tag{30}$$

In particular with the choice $s=s^\lambda$, by means of Eq. (22), one has that

$$\|Qs\|_{\mathcal{S}}=\left\|Pf^\lambda\right\|_{\mathcal{H}}\geq\left\|f^\lambda\right\|_{\mathcal{H}},$$

and, hence, that $Qs^\lambda=Pf^\lambda=f^\lambda$. Therefore, it follows that

$$I[s]+\lambda\|Qs\|_{\mathcal{S}}^2\geq I[s^\lambda]+\lambda\left\|Qs^\lambda\right\|_{\mathcal{S}}^2,$$

that is, s^λ is a minimizer of $I[s]+\lambda\|Ps\|_{\mathcal{S}}^2$.

Before proving the second part of the theorem we note that the following inequality follows as a simple consequence of the definition of projection.

$$I[s]+\lambda\|Qs\|_{\mathcal{S}}^2=I[f+g]+\lambda\|Pf\|_{\mathcal{H}}^2\leq I[f+g]+\lambda\|f\|_{\mathcal{H}}^2. \tag{31}$$

Assume now that $s^\lambda\in\mathcal{S}$ is a minimizer of $I[s]+\lambda\|Qs\|_{\mathcal{S}}^2$. Let $f^\lambda=Qs^\lambda$ and $g^\lambda=s-f^\lambda$, then, by Eq. (31) and Eq. (22), it follows that

$$I[f^\lambda+g^\lambda]+\lambda\left\|f^\lambda\right\|_{\mathcal{H}}^2\leq\inf_{(f,g)\in\mathcal{H}\times\mathcal{B}}\{I[f+g]+\lambda\|f\|_{\mathcal{H}}^2\}.$$

However, using Eq. (29) with $s=f^\lambda+g^\lambda$, one has that

$$I[f^\lambda+g^\lambda]+\lambda\left\|f^\lambda\right\|_{\mathcal{H}}^2\geq\inf_{(f,g)\in\mathcal{H}\times\mathcal{B}}\{I[f+g]+\lambda\|f\|_{\mathcal{H}}^2\}.$$

So $I[f^\lambda+g^\lambda]+\lambda\left\|f^\lambda\right\|_{\mathcal{H}}^2$ is the infimum of $I[f+g]+\lambda\|f\|_{\mathcal{H}}^2$ on $\mathcal{H}\times\mathcal{B}$. Clearly, if $g^\lambda\in\mathcal{B}$, it follows that (f^λ,g^λ) is a minimizer of $I[f+g]+\lambda\|f\|_{\mathcal{H}}^2$. ■

5.4 A Counterexample

The following example shows that in some pathological framework the minimization on $\mathcal{H}\times\mathcal{B}$ is not equivalent to the one on $\mathcal{S}=\mathcal{H}+\mathcal{B}$.

Example 1 *Let $\mathcal{H}=\ell_2=\{f=(f_n)_{n\in\mathbb{N}}\,|\,\sum_n f_n^2<+\infty\}$. The space ℓ_2 is a RKHS on $\mathbb{N}$ with respect to the kernel $K(n,m)=\delta_{n,m}$. Let $\mathcal{B}=\{f\in\ell_2\,|\,\sum_n n^2f_n^2<+\infty\}$ with the scalar product*

$$\langle f,g\rangle_{\mathcal{B}}=\sum_n n^2f_ng_n.$$

The space $\mathcal{B}$ is a RKHS with respect to the kernel $K^{\mathcal{B}}(n,m) = \frac{1}{n^2}\delta_{n,m}$.

Clearly, $\mathcal{B} \subset \mathcal{H}$, so that $\mathcal{H} \cap \mathcal{B} = \mathcal{B}$, which is not closed in $\mathcal{H}$. Since $\mathcal{B}$ is dense in $\mathcal{H}$, $P = 0$ and, by Lemma 7, $Q = 0$.

Let V be the squared loss function and choose $h = (h_n)_{n\in N} \in \mathcal{H}$ such that $h \notin \mathcal{B}$. Let $\rho(n,y) = \delta(y - h_n)$ so that

$$I[s] = \|s - h\|_{\mathcal{S}}^2,$$

then

$$I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 = \|s - h\|_{\mathcal{S}}^2,$$

and the minimizer is $s^\lambda = h$. Moreover, by our theorem, one has that

$$\inf_{f\in\mathcal{H}, g\in\mathcal{B}} \{I[f+g] + \lambda \|f\|_{\mathcal{H}}^2\} = I[s^\lambda] + \lambda \left\|Qs^\lambda\right\|_{\mathcal{S}}^2 = 0.$$

If $(f^\lambda, g^\lambda) \in \mathcal{H} \times \mathcal{B}$ were a minimizer, then $f^\lambda = 0$ and, hence, $g^\lambda = h$, but this is impossible since $h \notin \mathcal{B}$.

6. Existence and Uniqueness

We now discuss existence and uniqueness of the regularized solution in $\mathcal{S}$. Before stating and proving the main results we summarize our findings and show that if the offset space is empty both existence and uniqueness are easily obtained. Our analysis extends existence to all cases of interest under some weak assumptions on the kernel and the loss function for both regression and classification.

Uniqueness depends critically on the convexity assumption. For strictly convex functions we prove that the solution is unique if and only if the offset space satisfies suitable conditions, fulfilled in the case of constant offsets. For loss functions which are not strictly convex we limit our attention to the hinge loss and show that the solution is unique unless some particular conditions on the number and location of the support vectors are met. In Burges and Crisp (2000, 2003) similar results were obtained considering the dual formulation of the minimization problem.

If the offset space is empty, strict convexity and coerciveness of the penalty term trivially imply both existence and uniqueness. Indeed, we have the following proposition.

Proposition 8 *Given $\lambda > 0$, there exists a unique solution of the problem*

$$\min_{f\in\mathcal{H}} \left(I[f] + \lambda \|f\|_{\mathcal{H}}^2\right).$$

Proof The function $\left(I[f] + \lambda \|f\|_{\mathcal{H}}^2\right)$ is strictly convex and continuous. Moreover

$$I[f] + \lambda \|f\|_{\mathcal{H}}^2 \geq \lambda \|f\|_{\mathcal{H}}^2 \to +\infty$$

if $\|f\|_{\mathcal{H}}$ goes to $+\infty$. From item 4 of Proposition 14 both existence and uniqueness follow. ■

6.1 Existence

We now consider existence. If $\mathcal{B}$ is not trivial, there are no general results (see Wahba, 1990, for a discussion on this subject). However, if $\mathcal{B}$ is the set of constant functions, we derive existence of the solution in two different settings.

The first proposition holds only for classification under the assumption that the loss function V goes to infinity when $yf(\mathbf{x})$ goes to $-\infty$ (see Condition 1 of Proposition 9 below). Similar results were obtained in Steinwart (2002). We let ν be the marginal measure on X associated with ρ and $\operatorname{supp}\nu$ its support.

Proposition 9 *Assume that the following conditions hold*

1. $\lim_{w\to-\infty} V(1,w) = +\infty$ *and* $\lim_{w\to+\infty} V(-1,w) = +\infty$
2. *there is* $C > 0$ *such that* $\sqrt{K(\mathbf{x},\mathbf{x})} \leq C$ *for all* $\mathbf{x} \in \operatorname{supp}\nu$
3. $\rho(X \times \{1\}) > 0$ *and* $\rho(X \times \{-1\}) > 0$

Then there is at least one solution of the problem

$$\min_{s\in\mathcal{S}} \left(I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 \right),$$

where $\mathcal{S} = \mathcal{H} + \mathbb{R}$.

We observe that Assumption *2.* is satisfied if X is compact and K is continuous. Assumption *3.* has a very natural interpretation in the discrete setting where it simply amounts to have one example for each class. This condition is need since Assumption *1.* does not requires that V goes to $+\infty$ when $yf(\mathbf{x})$ goes to $+\infty$. Typical example of loss function satisfying Assumption *1.* is the hinge loss.

The second result holds both for regression and classification, but it requires the loss function going to infinity when $f(\mathbf{x})$ goes to $\pm\infty$, uniformly in y (compare Assumption *1.* of Proposition 10 and Assumption *1.* of Proposition 9).

Proposition 10 *Assume that the following conditions hold*

1. $\lim_{w\to\pm\infty}(\inf_{y\in Y} V(y,w)) = +\infty$.
2. *there is* $C > 0$ *such that* $\sqrt{K(\mathbf{x},\mathbf{x})} \leq C$ *for all* $\mathbf{x} \in \operatorname{supp}\nu$.

Then there is at least one solution of the problem

$$\min_{s\in\mathcal{S}} \left(I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 \right),$$

where $\mathcal{S} = \mathcal{H} + \mathbb{R}$.

We observe that for classification with symmetric loss functions, as the squared loss function, this proposition gives a sharper result than Proposition 9.

We now prove Proposition 9 and omit the proof of Proposition 10 since it is essentially the same.
Proof [of Proposition 9] The idea of the proof is to show that the functional we have to minimize goes to $+\infty$ when $\|s\|_{\mathcal{S}}$ goes to $+\infty$. With this aim, let

$$\alpha = \min\{\rho(X \times \{1\}), \rho(X \times \{-1\})\}.$$

By assumption *3*, $\alpha > 0$. For a fixed $M > 0$, we are looking for $R > 0$ such that for all $s \in \mathcal{S}$ with $\|s\|_{\mathcal{S}} \geq R$,

$$I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 \geq M.$$

Due to assumption *1*, there is $r > 0$ such that, for all $w \leq -r$, $V(1,w) \geq \frac{M}{\alpha}$ and, for all $w \geq r$, $V(-1,w) \geq \frac{M}{\alpha}$. We now let $R = \max\{2(1+C)\sqrt{\frac{M}{\lambda}}, 2r\}$ and choose $s \in \mathcal{S}$ with $\|s\|_{\mathcal{S}} \geq R$. If $\|Qs\|_{\mathcal{S}} = \|Qs\|_{\mathcal{H}} \geq \frac{R}{2(1+C)}$, then

$$\begin{aligned} I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 &\geq \lambda \|Qs\|_{\mathcal{S}}^2 \\ &\geq \lambda (\frac{R}{2(1+C)})^2 \\ &\geq M, \end{aligned}$$

since $R \geq 2(1+C)\sqrt{\frac{M}{\lambda}}$. If $\|Qs\| \leq \frac{R}{2(1+C)}$, let $b = s - Qs \in \mathbb{R}$, then

$$\begin{aligned} |b| &= \|s - Qs\|_{\mathcal{S}} \\ &\geq \|s\|_{\mathcal{S}} - \|Qs\|_{\mathcal{S}} \\ &\geq R - \frac{R}{2(1+C)} = R\frac{2C+1}{2C+2}. \end{aligned}$$

Assume, for example, that $b > 0$. For all $\mathbf{x} \in \operatorname{supp}\nu$

$$\begin{aligned} s(\mathbf{x}) &= \langle Qs, K_{\mathbf{x}} \rangle_{\mathcal{H}} + b \\ &\geq b - \|Qs\|_{\mathcal{H}} \|K_{\mathbf{x}}\|_{\mathcal{H}} \\ &\geq R\frac{2C+1}{2C+2} - \frac{R}{2(1+C)}C \\ &\geq R\frac{C+1}{2C+2} \\ &= \frac{R}{2} \geq r, \end{aligned}$$

since $R \geq \frac{r}{2}$. By definition of r, one has that for all $\mathbf{x} \in \operatorname{supp}\nu$

$$V(-1, s(\mathbf{x})) \geq \frac{M}{\alpha}.$$

Integrating both sides, we find

$$\int_{X \times \{-1\}} V(-1, s(\mathbf{x}))\, d\rho(\mathbf{x}, -1) \geq \frac{M}{\alpha} \rho(X \times \{-1\}) \geq M$$

from which it follows that

$$I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 \geq M.$$

The same proof holds when $b < 0$ replacing the integration on $X \times \{-1\}$ with the integration on $X \times \{1\}$. Since M is arbitrary, we have that

$$I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 \geq \lambda \|Qs\|_{\mathcal{S}}^2 \to +\infty.$$

Since the functional is continuous, from item 4 of Proposition 14 the existence of the minimizer follows. ■

6.2 Uniqueness

The first proposition completely characterizes uniqueness for strictly convex functions.

Proposition 11 *Let s^λ be a solution of the problem*

$$\min_{s\in\mathcal{S}} \left(I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 \right).$$

1. *If s' is another solution, then $Qs' = Qs^\lambda$.*
2. *If $V(y,\cdot)$ is strictly convex for all $y \in Y$ then all the minimizers are of the form $s^\lambda + g$, with $g \in \mathcal{S}$ such that $Qg = 0$ and $g(\mathbf{x}) = 0$ for ν-almost all $x \in X$.*

Let us comment on this proposition before providing the proof. We recall that a solution s^λ is the sum of two terms: $f^\lambda = Qs^\lambda$ which is orthogonal to $\mathcal{B}$ and $g^\lambda = s^\lambda - f^\lambda$. The uniqueness of f^λ (item *1*) is due to the strict convexity of the penalty term. Item *2* states the general conditions that should be satisfied by offset functions to obtain uniqueness on s^λ: in the discrete setting one has uniqueness if and only if the condition $g(\mathbf{x}_i) = 0$ for all i implies that g is equal to zero. Clearly, if $\mathcal{B}$ is the space of constant functions uniqueness is ensured. We now give the proof of the proposition.

Proof [of Proposition 11]

1. Let s' another minimizer and assume that $Qs^\lambda \neq Qs'$. Then, by the strict convexity of $\|\cdot\|_{\mathcal{S}}^2$, one has that, for all $t \in]0,1[$,

$$\left\| (1-t)Qs^\lambda + tQs' \right\|_{\mathcal{S}}^2 < (1-t) \left\| Qs^\lambda \right\|_{\mathcal{S}}^2 + t \left\| Qs' \right\|_{\mathcal{S}}^2.$$

   Since $I[s]$ is convex, one has that

$$I[(1-t)s^\lambda + ts'] \leq (1-t)I[s^\lambda] + tI[s'].$$

   From the above two inequalities we find

$$\begin{aligned} I[(1-t)s^\lambda + ts'] &+ \lambda \left\| Q\left((1-t)s^\lambda + ts' \right) \right\|_{\mathcal{S}}^2 \\ &< (1-t)\left(I[s^\lambda] + \lambda \left\| Qs^\lambda \right\|_{\mathcal{S}}^2 \right) + t\left(I[s'] + \lambda \left\| Qs' \right\|_{\mathcal{S}}^2 \right) \\ &= \min_{s\in\mathcal{S}} \left(I[s] + \lambda \|Qs\|_{\mathcal{S}}^2 \right). \end{aligned}$$

   Since this is impossible, it follows that $Qs^\lambda = Qs'$.

2. Let $s' = s^\lambda + g$ with g as in item *1*. By straightforward computation we have that s' is a minimizer. It is left to show that the minimizers are only the functions written in the above form. From item *1* we have that $Qg = 0$. Let U be the measurable set

$$U = \{\mathbf{x} \in X \,|\, g(\mathbf{x}) \neq 0\} = \{\mathbf{x} \in X \,|\, s'(\mathbf{x}) \neq s^\lambda(\mathbf{x})\}.$$

   By contradiction, let us assume that $\nu(U) > 0$ and, hence, $\rho(U \times Y) > 0$. Fix $t \in]0,1[$. since $V(y,\cdot)$ is strictly convex, for all $(\mathbf{x},y) \in U \times Y$, one has that

$$V(y,(1-t)s^\lambda(\mathbf{x}) + ts'(\mathbf{x})) < (1-t)V(y,s^\lambda(\mathbf{x})) + tV(y,s'(\mathbf{x})).$$

Therefore, by integration,

$$\int_{U\times Y} V(y,(1-t)s^\lambda(\mathbf{x})+ts'(\mathbf{x}))\,d\rho(\mathbf{x},y) <$$

$$< (1-t)\int_{U\times Y} V(y,s^\lambda(\mathbf{x}))\,d\rho(\mathbf{x},y)+t\int_{U\times Y} V(y,s'(\mathbf{x}))\,d\rho(\mathbf{x},y).$$

On the complement of $U\times Y$, we have $V(y,s^\lambda(\mathbf{x}))=V(y,s'(\mathbf{x}))$, so that

$$I[(1-t)s^\lambda+ts'] < (1-t)I[s^\lambda]+tI[s'].$$

By the same line of reasoning of item *1*, one finds a contradiction. It follows that $\nu(U)=0$, that is, $g(\mathbf{x})=0$ for ν-almost all $\mathbf{x}\in X$.

■

Two important examples of convex loss functions which are not strictly convex are the hinge and the ε-insensitive loss. The next proposition deals with the hinge loss though a similar result can be also derived for the ε-insensitive loss, see Burges and Crisp (2000). For the sake of simplicity we develop our result in the discrete setting for the case of constant offset functions. In this case uniqueness of the solution is expressed as a condition on the number of support vectors of the two classes. Similar but a little bit more involved conditions can be found considering the continuous setting.

Proposition 12 *Let $Y=\{\pm 1\}$, $V(y,w)=|1-yw|_+$ and $\mathcal{B}=\mathbb{R}$. Let s^λ be a solution of*

$$\min_{s\in\mathcal{S}}\left(\frac{1}{\ell}\sum_{i=1}^{\ell} V(y_i,s(\mathbf{x}_i))+\lambda\|Qs\|_{\mathcal{S}}^2\right),$$

and define

$$I_+=\{i\,|\,y_i=1,\,s^\lambda(\mathbf{x})<1\} \qquad I_-=\{i\,|\,y_i=-1,\,s^\lambda(\mathbf{x})>-1\}$$
$$B_+=\{i\,|\,y_i=1,\,s^\lambda(\mathbf{x}_1)=1\} \qquad B_-=\{i\,|\,y_i=-1,\,s^\lambda(\mathbf{x}_1)=-1\}.$$

The solution is unique if and only if

$$\#I_+\neq\#I_-+\#B_- \tag{32}$$

and

$$\#I_-\neq\#I_++\#B_+, \tag{33}$$

where # denotes set cardinality.

Proof Assume that s' is another solution. From item *1* of proposition 11, we have that $Qs^\lambda=Qs'$ and $s'=s^\lambda+b$. Since both functions are minimizers, one concludes that

$$\sum_{i=1}^{\ell}|1-y_is^\lambda(\mathbf{x}_i)|_+=\sum_{i=1}^{\ell}|1-y_is'(\mathbf{x}_i)|_+ \tag{34}$$

We notice that if $yw_1 < 1$ and $yw_2 > 1$, then

$$V(y,(1-t)w_1+tw_2) < (1-t)V(y,w_1)+tV(y,w_2).$$

Reasoning as in the proof of the previous proposition, one has that, for all $i \in I_+ \cup I_-$,

$$y_i s'(\mathbf{x}_i) \leq 1$$

and, for all $i \notin (I_+ \cup I_- \cup B_+ \cup B_-)$

$$y_i s'(\mathbf{x}_i) \geq 1.$$

Using the above two equations, it follows that equality (34) becomes

$$\sum_{i \in I_+ \cup I_-} (1 - y_i s^\lambda(\mathbf{x}_i)) = \sum_{i \in I_+ \cup I_-} (1 - y_i s'(\mathbf{x}_i)) + \sum_{i \in B_+ \cup B_-} |-by_i|_+,$$

(if the index set is empty, we let the corresponding sum be equal to 0). The above equation is equivalent to

$$\sum_{i \in I_+ \cup I_-} by_i = \sum_{i \in B_+ \cup B_-} |-by_i|_+,$$

that has a not trivial solution if and only if both the following conditions are true

1. if $b > 0$, then $\sum_{i \in I_+ \cup I_-} y_i = -\sum_{B_-} y_i$ (that is, Eq. (32) holds).
2. if $b < 0$, then $\sum_{i \in I_+ \cup I_-} y_i = \sum_{B_+} y_i$ (that is, Eq. (33) holds).

Now, if neither Eq. (32) nor Eq. (33) holds, then $b = 0$ and s^λ is unique. Conversely, assume for example that Eq. (32) holds. It is simple to check that there is $b > 0$ such that for all $i \in I_+ \cup I_-$,

$$y_i(s^\lambda(\mathbf{x}_i) + b) \leq 1$$

and, for all $i \notin (I_+ \cup I_- \cup B_+ \cup B_-)$

$$y_i(s^\lambda(\mathbf{x}_i) + b) \geq 1.$$

Finally, by direct computation one has that

$$I[s^\lambda] = I[s^\lambda + b].$$

■

If the solution is not unique, the solution family is parameterized as $s^\lambda + b$, where b runs in a closed, not necessarily bounded interval. However, if there is at least one example for each class, b lies in the bounded interval $[b_-, b_+]$ and one can easily show that

1. for the solution with $b = b_-$, Eq. (32) holds;
2. for the solution with $b = b_+$, Eq. (33) holds;
3. for the solution with $b_- < b < b_+$, both Eqs. (32) and (33) hold, from which it follows that $\#I_+ = \#I_-$ and $\#B_+ = \#B_- = 0$.

7. Discrete Tikhonov Regularization

We now specialize our results to the case in which the probability measure is the empirical distribution ρ_S and $\mathcal{B}$ is the space of constant functions ($K^{\mathcal{B}} = 1$) and discuss in detail Support Vector Machines for classification.

We start by recalling that, from item 2 of Proposition 14 it follows that the left and right derivatives of $V(y,\cdot)$ always exist and

$$(\partial V)(y,w) = [V'_-(y,w), V'_+(y,w)].$$

Corollary 13 *Let $\mathcal{S} = \mathcal{H} + \mathbb{R}$ and Q the projection on*

$$\{s \in \mathcal{S} \mid \langle s, 1\rangle_{\mathcal{S}} = 0\}.$$

Given $\lambda > 0$, let $f^\lambda \in \mathcal{H}$ and $b^\lambda \in \mathbb{R}$ and define $s^\lambda = f^\lambda + b^\lambda \in \mathcal{S}$, then

$$(f^\lambda, b^\lambda) \in \operatorname*{argmin}_{f\in\mathcal{H}, b\in\mathbb{R}} \left\{ \frac{1}{\ell}\sum_i V(y_i, f(\mathbf{x}_i)+b) + \lambda \|f\|^2_{\mathcal{H}} \right\}$$

if and only if

$$\begin{aligned} s^\lambda &\in \operatorname*{argmin}_{s\in\mathcal{S}} \left\{ \frac{1}{\ell}\sum_i V(y_i, s(\mathbf{x}_i)) + \lambda \|Qs\|^2_{\mathcal{H}} \right\} \\ f^\lambda &= Qs^\lambda \end{aligned}$$

if and only if there are $\alpha_1, \ldots, \alpha_\ell \in \mathbb{R}$ such that

$$\begin{aligned} f^\lambda &= \sum_{i=1}^\ell \alpha_i K_{\mathbf{x}_i} = \sum_{i=1}^\ell \alpha_i (K_{\mathbf{x}_i} + 1) \\ \frac{-1}{2\lambda\ell} V'_+(y_i, f^\lambda(\mathbf{x}_i)+b^\lambda) \le \alpha_i &\le \frac{-1}{2\lambda\ell} V'_-(y_i, f^\lambda(\mathbf{x}_i)+b^\lambda) \\ \sum_{i=1}^\ell \alpha_i &= 0 \end{aligned}$$

We notice two facts. First, α_i can be zero only if $0 \in (\partial V)(y_i, f^\lambda(\mathbf{x}_i)+b^\lambda)$ – that is, only if $f^\lambda(\mathbf{x}_i) + b^\lambda$ is a minimizer of $V(y_i,\cdot)$. Therefore, a necessary condition for obtaining sparsity is a *plateaux* in the loss function. A quantitative discussion on this topic can be found in Steinwart (2003). Second if V_- and V_+ are bounded by a constant $M > 0$, one has that $|\alpha_i| \le 2\lambda\ell M$ – that is, a sufficient conditions for box constraints on the coefficients.

In the rest of this section we consider Support Vector Machines for classification showing that through our analysis the solution is completely characterized in the primal formulation.

A simple calculation for the hinge loss shows that

$$[V'_-(y,w), V'_+(y,w)] = \begin{cases} -y & \text{for } yw < 1 \\ [\min\{-y,0\}, \max\{0,-y\}] & \text{for } yw = 1 \\ 0 & \text{for } yw > 1 \end{cases}. \tag{35}$$

To be consistent with the notation used in the literature, we let $C = \frac{1}{2\lambda\ell}$ and factorize the labels y_i from the coefficients α_i. Then, according to the above corollary, the solution of the SVM algorithm is given by

$$s^\lambda = \sum_{i=1}^{\ell} \alpha_i y_i K_{\mathbf{x}_i} + b^\lambda$$

where the set $(\alpha_1, \ldots, \alpha_\ell, b^\lambda)$ solves the following algebraic system of inequalities

$$\begin{aligned} 0 \le \alpha_i \le C \quad &\text{if} \quad y_i \left(\sum_{j=1}^{\ell} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + b^\lambda \right) = 1 \\ \alpha_i = 0 \quad &\text{if} \quad y_i \left(\sum_{j=1}^{\ell} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + b^\lambda \right) > 1 \\ \alpha_i = C \quad &\text{if} \quad y_i \left(\sum_{j=1}^{\ell} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + b^\lambda \right) < 1 \\ \sum_i \alpha_i y_i \quad &= \quad 0 \end{aligned} \tag{36}$$

Interestingly, the above inequalities, which fully characterize the support vectors associated with the solution, are usually obtained as the Kuhn-Tucker conditions of the dual QP optimization problem (Vapnik, 1988).

Looking at Eqs.(35-36), it is immediate to see that the box constraints ($0 \le \alpha_i \le C$) are due to the linearity of $V(yf(\mathbf{x}))$ for $yf(\mathbf{x}) < 1$, whereas sparsity ($\alpha_i = 0$) follows from the constancy of $V(yf(\mathbf{x}))$ for $yf(\mathbf{x}) > 1$.

8. Conclusion

In this paper we study some properties of learning functionals derived from Tikhonov regularization. We develop our analysis in a continuous setting and use tools from convex analysis in infinite dimensional spaces to quantitatively characterize the explicit form of the regularized solution for both regression and classification. We also address the case with and without the offset term within the same unifying framework. We show that the presence of an offset term is equivalent to solving a standard problem of regularization in a Reproducing Kernel Hilbert Space in which the penalty term is given by a seminorm. Finally, we discuss issues of existence and uniqueness of the solution and specialize our results to the discrete setting.

Current work aims at extending these results to vector-valued functions (Micchelli and Pontil, 2003) and exploring possible use of offset functions to incorporate invariances (Girosi and Chan, 1995).

Acknowledgments

We thank the anonymous referees for suggestions leading to an improved version of the paper. A. Caponnetto is supported by a PRIN fellowship within the project “Inverse problems in medical imaging”, n. 2002013422. This research has been partially funded by the INFM Project MAIA,

the FIRB Project ASTAA and the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778.

Appendix A. Convex Functions in Infinite Dimensional Spaces

The proof of Theorem 2 is based on the properties of convex functions defined on infinite dimensional spaces. In particular, we use the notion of subgradient that extends the notion of derivative to convex non-differentiable functions. In this appendix we collect the results we need. For details see the book Ekeland and Turnbull (1983) and also Ekeland and Teman (1974).

Let $\mathcal{H}$ be a Banach space and $\mathcal{H}^*$ its dual. A function $F : \mathcal{H} \to \mathbb{R} \cup +\infty$ is *convex* if

$$F(tv+(1-t)w) \leq tF(v)+(1-t)F(w),$$

for all $v,w \in \mathcal{H}$ and $t \in [0,1]$ (if the strict inequality holds for $t \in (0,1)$, F is called *strictly convex*).

Let $v_0 \in \mathcal{H}$ such that $F(v_0) < +\infty$. The *subgradient* of F at point $v_0 \in \mathcal{H}$ is the subset of $\mathcal{H}^*$ given by

$$\partial F(v_0) = \{w \in \mathcal{H}^* \,|\, F(v) \geq F(v_0) + \langle w, v-v_0 \rangle \,,\ \forall v \in \mathcal{H}\}. \tag{37}$$

where $\langle \cdot, \cdot \rangle$ is the pairing between $\mathcal{H}^*$ and $\mathcal{H}$. If $F(v) = +\infty$, we let $\partial F(v_0) = \emptyset$.

In the following proposition we summarize the main properties of the subgradient we need.

Proposition 14 *The following facts hold:*

1. *If F is differentiable at v_0, the subgradient reduces to the usual gradient $F'(v_0)$.*
2. *If F is defined on $\mathbb{R}$ and $F(v_0) < +\infty$, then F admits left and right derivative and*

$$\partial F(v_0) = [F'_-(v_0), F'_+(v_0)].$$

3. *Assume that $F \neq +\infty$. A point v_0 is a minimizer of F if and only if $0 \in \partial F(v_0)$.*
4. *If F is continuous and*

$$\lim_{\|v\|_{\mathcal{H}} \to +\infty} F(v) = +\infty.$$

   then F has a minimizer. If F is strictly convex, the minimizer is unique.

5. *Let G be another convex function on $\mathcal{H}$. Assume that there is $v_0 \in \mathcal{H}$ such that F and G are continuous and finite at v_0. Let $a,b \geq 0$, then $aF+bG$ is convex and, for all $v \in \mathcal{H}$,*

$$\partial(aF+bG)(v) = a(\partial F)(v) + b(\partial G)(v).$$

6. *Let $\mathcal{H}'$ be another Banach space and $\mathcal{J}$ be a continuous linear operator from $\mathcal{H}'$ into $\mathcal{H}$. Assume that there is $v'_0 \in \mathcal{H}'$ such that F is continuous and finite at $\mathcal{J}v'_0$. For all $v' \in \mathcal{H}'$*

$$(\partial F \circ \mathcal{J})(v') = \mathcal{J}^*(\partial F)(\mathcal{J}v'),$$

   where $\mathcal{J}^* : \mathcal{H}^* \to \mathcal{H}'^*$ is the adjoint of $\mathcal{J}$ defined by

$$\left\langle v', \mathcal{J}^* v \right\rangle_{\mathcal{H}'} = \left\langle \mathcal{J}v', v \right\rangle_{\mathcal{H}}.$$

   for all $v \in \mathcal{H}$ and $v' \in \mathcal{H}'$.

Proof We simply give the references to the book of Ekeland and Turnbull (1983).

1. Prop. III.2.8
2. Prop. III.2.7
3. It is a simple consequence of Prop. III.3.1
4. It is a simple consequence of Prop. II.4.6.
5. Prop. III.2.13
6. Prop. III.2.12

■

We now recall the definition of *Nemitski* functional, adapted to our framework (Ekeland and Turnbull, 1983, p.143). Let Z be a locally compact second countable space, ρ be a finite measure on Z, and $W : Z \times \mathbb{R} \to [0, +\infty[$ be a measurable function on $Z \times \mathbb{R}$ such that $W(z, \cdot)$ is convex for all $z \in Z$ (since $W(z, \cdot)$ is convex on $\mathbb{R}$, it is continuous).

Let $p \in [1, +\infty[$ and $L^p(Z, \rho)$ be the Banach space of measurable functions $u : Z \to \mathbb{R}$ such that $\int_Z |u(z)|^p \, d\rho(z)$ is finite.

The *Nemitski* functional associated with W is defined as the map $I_0 : L^p(X, \nu) \to [0, +\infty[\cup \{+\infty\}$ given by

$$I_0[u] = \int_Z W(z, u(z)) d\rho(z). \tag{38}$$

Next proposition provides us with a straightforward method to study the subgradient (∂I_0). Let $q \in]1, +\infty]$ such that $\frac{1}{p} + \frac{1}{q} = 1$.

Proposition 15 *Assume that there is an element* $u_0 \in L^p(Z, \nu)$ *such that* $\sup_{z \in Z} |u_0(z)| < +\infty$ *and* $I_0[u_0] < +\infty$. *Given* $u \in L^p(Z, \rho)$

$$(\partial I_0)(u) = \{w \in L^q(Z, \rho) \mid \ w(z) \in (\partial W)(z, u(z)) \ \rho - \text{a.e.}\}. \tag{39}$$

Proof See the proof of Prop. III.5.3 of Ekeland and Turnbull (1983). The proof is for Z interval of $\mathbb{R}$, but can be easily extended to arbitrary Z, compare with Ekeland and Teman (1974). ■

References

Peter L. Bartlett. Prediction algorithms: complexity, concentration and convexity. In *Proceedings of the 13th IFAC Symposium on System Identification*, pages 1507–1517, 2003.

Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local Rademacher complexities. (submitted), 2002.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. Technical Report 638, Department of Statistics, U. C. Berkeley, 2003.

M. Bertero, T. Poggio, and V. Torre. Ill-posed problems in early vision. *Proceedings of the IEEE*, 76:869–889, 1988.

C. Burges and D. Crisp. Uniqueness of the SVM solution. In *Proceedings of the Twelfth Conference on Neural Information Processing Systems*, pages 223–229, Cambridge, MA, 2000. MIT Press.

C. Burges and D. Crisp. Uniqueness theorems for kernel methods. *Neurocomputing*, 55:187–220, 2003.

D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Ann. Stat.*, 18:1676–1695, 1990.

N. Cristianini and J. Shawe Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

F. Cucker and S. Smale. Best choices for regularization parameters in learning theory: on the bias-variance problem. *Foundations of Computationals Mathematics*, 2:413–428, 2002.

I. Ekeland and R. Teman. *Analyse convexe et problémes variationnels*. Gauthier-Villards, Paris, 1974.

I. Ekeland and T. Turnbull. *Infinite-dimensional Optimization and Convexity*. Chicago Lectures in Mathematics. The University of Chicago Press, Chicago, 1983.

T. Evgeniou, Pontil M., and T. Poggio. Regularization networks and support vector machines. *Adv. Comp. Math.*, 13:1–50, 2000.

F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10:1455–1480, 1998.

F. Girosi and N. Chan. Prior knowledge and the creation of virtual examples for rbf networks. In *Proceedings of the IEEE-SP Workshop on Neural Networks Signal Processing*, pages 201–210, Cambridge, MA, 1995. IEEE Signal Processing Society.

G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Ann. Math. Stat.*, 41:495–502, 1970.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Research Note RN/03/08, Dept of Computer Science, UCL*, 2003.

S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. Statistical learning: Stability is sufficient for generalization and necessary and sufficient for consistency for empirical risk minimization. C.B.C.L. Paper 223, Massachusetts Institute of Technology - Artificial Intelligence Laboratory, December 2002.

P. Niyogi and F. Girosi. Generalization bounds for function approximation from scattered data. *Adv. Comp. Math.*, 10:51–80, 1999.

T. Poggio and F. Girosi. A theory of networks for approximation and learning. In C. Lau, editor, *Foundation of Neural Networks*, pages 91–106. IEEE Press, Piscataway, New Jersey, 1992.

T. Poggio, S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri. B. In J. Winkler and M. Niranjan, editors, *Uncertainty in Geometric Computations*, pages 131–141. Kluwer Academic Publishers, 2002.

T. Poggio and S. Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50:537–544, 2003.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In D. Helmbold and B. Williamson, editors, *Neural Networks and Computational Learning Theory*, number 81, pages 416–426. Springer, Berlin, Germany, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002. URL `http://www.learning-with-kernels.org`.

L. Schwartz. Sous-espaces hilbertiens d'espaces vectorials topologiques and noyaux associès. *Journal d'Analyse Mathematique*, 13:115–256, 1964.

I. Steinwart. Consistency of support vector machines and other regularized kernel machines. *submitted to IEEE Transactions on Information Theory*, 2002.

I. Steinwart. Sparseness of support vector machines. *Journal of Machine Learning Research*, 4: 1071–1105, 2003.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill Posed Problems*. W. H. Winston, Washington, D. C., 1977.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1988.

G. Wahba. *Splines Models for Observational Data*, volume 59 of *Series in Applied Mathematics*. SIAM, 1990.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and randomized GACV. Technical Report 984, Department of Statistics, University of Wisconsin, 1998.

T. Zhang. Convergence of large margin separable linear classification. In T. G. Len, T. K. Dietterich and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 357–363. MIT Press, 2001.

Journal of Machine Learning Research 5 (2004) 1391–1415 Submitted 3/04; Published 10/04

The Entire Regularization Path for the Support Vector Machine

Trevor Hastie HASTIE@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Saharon Rosset SROSSET@US.IBM.COM
IBM Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598, USA

Robert Tibshirani TIBS@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Ji Zhu JIZHU@UMICH.EDU
Department of Statistics
University of Michigan
439 West Hall
550 East University
Ann Arbor, MI 48109-1092, USA

Editor: Nello Cristianini

Abstract

The support vector machine (SVM) is a widely used tool for classification. Many efficient implementations exist for fitting a two-class SVM model. The user has to supply values for the tuning parameters: the regularization cost parameter, and the kernel parameters. It seems a common practice is to use a default value for the cost parameter, often leading to the least restrictive model. In this paper we argue that the choice of the cost parameter can be critical. We then derive an algorithm that can fit the entire path of SVM solutions for every value of the cost parameter, with essentially the same computational cost as fitting one SVM model. We illustrate our algorithm on some examples, and use our representation to give further insight into the range of SVM solutions.

Keywords: support vector machines, regularization, coefficient path

1. Introduction

In this paper we study the support vector machine (SVM)(Vapnik, 1996; Schölkopf and Smola, 2001) for two-class classification. We have a set of n training pairs x_i, y_i, where $x_i \in \mathbb{R}^p$ is a p-vector of real-valued predictors (attributes) for the ith observation, and $y_i \in \{-1,+1\}$ codes its binary response. We start off with the simple case of a linear classifier, where our goal is to estimate a linear decision function

$$f(x) = \beta_0 + \beta^T x, \tag{1}$$

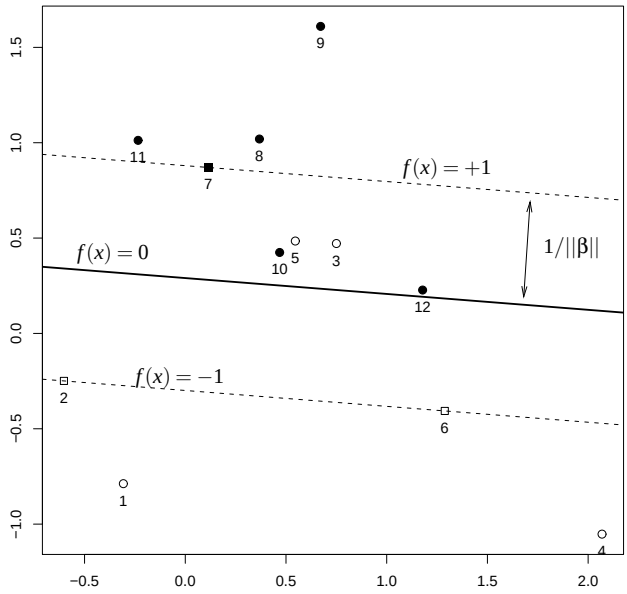


Figure 1: A simple example shows the elements of a SVM model. The "+1" points are solid, the "-1" hollow. $C = 2$, and the width of the soft margin is $2/||\beta|| = 2 \times 0.587$. Two hollow points $\{3,5\}$ are misclassified, while the two solid points $\{10,12\}$ are correctly classified, but on the wrong side of their margin $f(x) = +1$; each of these has $\xi_i > 0$. The three square shaped points $\{2,6,7\}$ are exactly on the margin.

and its associated classifier

$$\text{Class}(x) = \text{sign}[f(x)]. \tag{2}$$

There are many ways to fit such a linear classifier, including linear regression, Fisher's linear discriminant analysis, and logistic regression (Hastie et al., 2001, Chapter 4). If the training data are linearly separable, an appealing approach is to ask for the decision boundary $\{x : f(x) = 0\}$ that maximizes the margin between the two classes (Vapnik, 1996). Solving such a problem is an exercise in convex optimization; the popular setup is

$$\min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 \text{ subject to, for each i: } y_i(\beta_0 + x_i^T\beta) \geq 1. \tag{3}$$

A bit of linear algebra shows that $\frac{1}{||\beta||}(\beta_0 + x_i^T\beta)$ is the signed distance from x_i to the decision boundary. When the data are not separable, this criterion is modified to

$$\begin{aligned} \min_{\beta_0,\beta} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n}\xi_i, & \qquad (4) \\ \text{subject to, for each } i\text{: } y_i(\beta_0 + x_i^T\beta) \quad &\geq \quad 1 - \xi_i. \end{aligned}$$

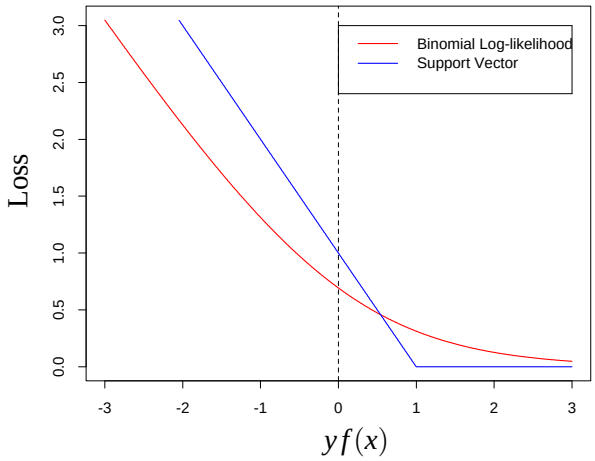


Figure 2: The hinge loss penalizes observation margins $yf(x)$ less than $+1$ linearly, and is indifferent to margins greater than $+1$. The negative binomial log-likelihood (deviance) has the same asymptotes, but operates in a smoother fashion near the *elbow* at $yf(x) = 1$.

Here the ξ_i are non-negative slack variables that allow points to be on the wrong side of their "soft margin" ($f(x) = \pm 1$), as well as the decision boundary, and C is a cost parameter that controls the amount of overlap. Figure 1 shows a simple example. If the data are separable, then for sufficiently large C the solutions to (3) and (4) coincide. If the data are not separable, as C gets large the solution approaches the minimum overlap solution with largest margin, which is attained for some finite value of C.

Alternatively, we can formulate the problem using a *Loss* + *Penalty* criterion (Wahba et al., 2000; Hastie et al., 2001):

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} [1 - y_i(\beta_0 + \beta^T x_i)]_+ + \frac{\lambda}{2}||\beta||^2. \tag{5}$$

The regularization parameter λ in (5) corresponds to $1/C$, with C in (4). Here the *hinge loss* $L(y, f(x)) = [1 - yf(x)]_+$ can be compared to the negative binomial log-likelihood $L(y, f(x)) = \log[1 + \exp(-yf(x))]$ for estimating the linear function $f(x) = \beta_0 + \beta^T x$; see Figure 2.

This formulation emphasizes the role of regularization. In many situations we have sufficient variables (e.g. gene expression arrays) to guarantee separation. We may nevertheless avoid the maximum margin separator ($\lambda \downarrow 0$), which is governed by observations on the boundary, in favor of a more regularized solution involving more observations.

This formulation also admits a class of more flexible, nonlinear generalizations

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda J(f), \tag{6}$$

where $f(x)$ is an arbitrary function in some Hilbert space $\mathcal{H}$, and $J(f)$ is a functional that measures the "roughness" of f in $\mathcal{H}$.

The nonlinear *kernel* SVMs arise naturally in this context. In this case $f(x) = \beta_0 + g(x)$, and $J(f) = J(g)$ is a norm in a Reproducing Kernel Hilbert Space of functions $\mathcal{H}_K$ generated by a

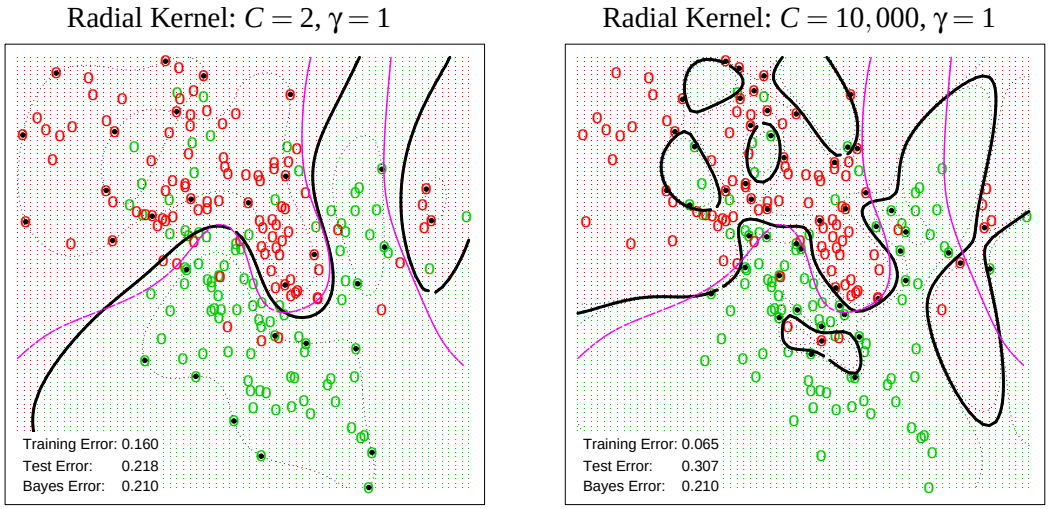


Figure 3: Simulated data illustrate the need for regularization. The 200 data points are generated from a pair of mixture densities. The two SVM models used radial kernels with the scale and cost parameters as indicated at the top of the plots. The thick black curves are the decision boundaries, the dotted curves the margins. The less regularized fit on the right overfits the training data, and suffers dramatically on test error. The broken purple curve is the optimal Bayes decision boundary.

positive-definite kernel $K(x,x')$. By the well-studied properties of such spaces (Wahba, 1990; Evgeniou et al., 1999), the solution to (6) is finite dimensional (even if $\mathcal{H}_K$ is infinite dimensional), in this case with a representation $f(x) = \beta_0 + \sum_{i=1}^{n} \theta_i K(x,x_i)$. Consequently (6) reduces to the finite form

$$\min_{\beta_0,\theta} \sum_{i=1}^{n} L[y_i, \beta_0 + \sum_{j=1}^{n} \theta_i K(x_i,x_j)] + \frac{\lambda}{2} \sum_{j=1}^{n} \sum_{j'=1}^{n} \theta_j \theta_{j'} K(x_j,x'_j). \tag{7}$$

With L the hinge loss, this is an alternative route to the kernel SVM; see Hastie et al. (2001) for more details.

It seems that the regularization parameter C (or λ) is often regarded as a genuine "nuisance" in the community of SVM users. Software packages, such as the widely used *SVM*^{light} (Joachims, 1999), provide default settings for C, which are then used without much further exploration. A recent introductory document (Hsu et al., 2003) supporting the `LIBSVM` package does encourage grid search for C.

Figure 3 shows the results of fitting two SVM models to the same simulated data set. The data are generated from a pair of mixture densities, described in detail in Hastie et al. (2001, Chapter 2).[1] The radial kernel function $K(x,x') = \exp(-\gamma||x-x'||^2)$ was used, with $\gamma = 1$. The model on the left is more regularized than that on the right ($C = 2$ vs $C = 10,000$, or $\lambda = 0.5$ vs $\lambda = 0.0001$), and

1. The actual training data and test distribution are available from `http:// www-stat.stanford.edu/ElemStatLearn`.

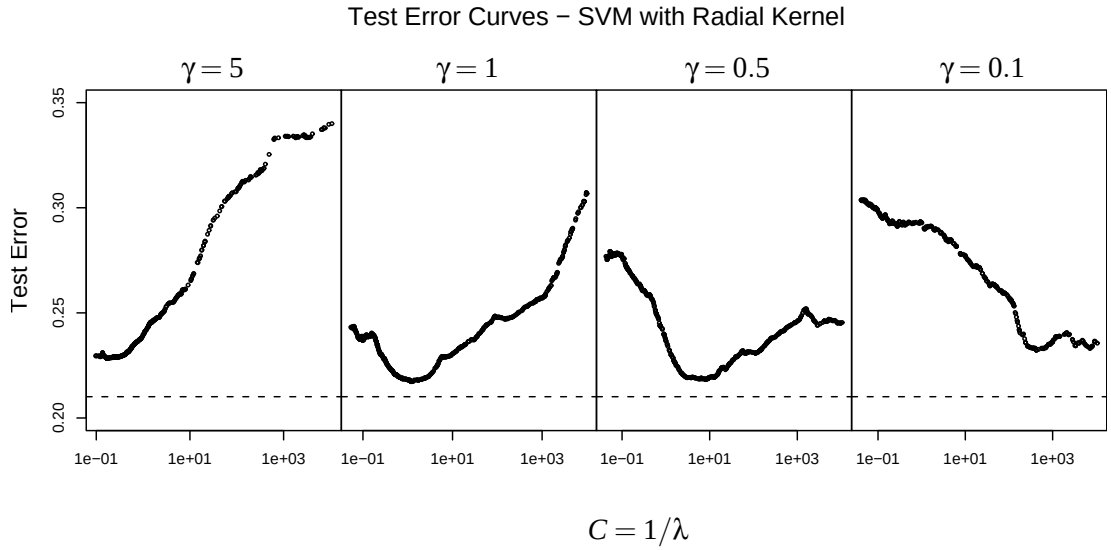


Figure 4: Test error curves for the mixture example, using four different values for the radial kernel parameter γ. Small values of C correspond to heavy regularization, large values of C to light regularization. Depending on the value of γ, the optimal C can occur at either end of the spectrum or anywhere in between, emphasizing the need for careful selection.

performs much better on test data. For these examples we evaluate the test error by integration over the lattice indicated in the plots.

Figure 4 shows the test error as a function of C for these data, using four different values for the kernel scale parameter γ. Here we see a dramatic range in the correct choice for C (or $\lambda = 1/C$); when $\gamma = 5$, the most regularized model is called for, and we will see in Section 6 that the SVM is really performing kernel density classification. On the other hand, when $\gamma = 0.1$, we would want to choose among the least regularized models.

One of the reasons that investigators avoid extensive exploration of C is the computational cost involved. In this paper we develop an algorithm which fits the *entire path* of SVM solutions $[\beta_0(C), \beta(C)]$, for all possible values of C, with essentially the computational cost of fitting a single model for a particular value of C. Our algorithm exploits the fact that the Lagrange multipliers implicit in (4) are piecewise-linear in C. This also means that the coefficients $\beta(C)$ are also piecewise-linear in C. This is true for all SVM models, both linear and nonlinear kernel-based SVMs. Figure 8 on page 1406 shows these Lagrange paths for the mixture example. This work was inspired by the related "Least Angle Regression" (LAR) algorithm for fitting LASSO models (Efron et al., 2004), where again the coefficient paths are piecewise linear.

These speedups have a big impact on the estimation of the accuracy of the classifier, using a validation dataset (e.g. as in K-fold cross-validation). We can rapidly compute the fit for each test data point for any and all values of C, and hence the generalization error for the entire validation set as a function of C.

In the next section we develop our algorithm, and then demonstrate its capabilities on a number of examples. Apart from offering dramatic computational savings when computing multiple solu-

tions (Section 4.3), the nature of the path, in particular at the boundaries, sheds light on the action of the kernel SVM (Section 6).

2. Problem Setup

We use a criterion equivalent to (4), implementing the formulation in (5):

$$\min_{\beta,\beta_0} \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\beta^T\beta \tag{8}$$
$$\text{subject to } 1 - y_i f(x_i) \le \xi_i;\ \xi_i \ge 0;\ f(x) = \beta_0 + \beta^T x.$$

Initially we consider only linear SVMs to get the intuitive flavor of our procedure; we then generalize to kernel SVMs.

We construct the Lagrange primal function

$$L_P: \qquad \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\beta^T\beta + \sum_{i=1}^{n} \alpha_i(1 - y_i f(x_i) - \xi_i) - \sum_{i=1}^{n} \gamma_i\xi_i \tag{9}$$

and set the derivatives to zero. This gives

$$\frac{\partial}{\partial\beta}: \qquad \beta = \frac{1}{\lambda}\sum_{i=1}^{n} \alpha_i y_i x_i, \tag{10}$$

$$\frac{\partial}{\partial\beta_0}: \qquad \sum_{i=1}^{n} y_i\alpha_i = 0, \tag{11}$$

$$\frac{\partial}{\partial\xi_i}: \qquad \alpha_i = 1 - \gamma_i, \tag{12}$$

along with the KKT conditions

$$\alpha_i(1 - y_i f(x_i) - \xi_i) = 0, \tag{13}$$
$$\gamma_i\xi_i = 0. \tag{14}$$

We see that $0 \le \alpha_i \le 1$, with $\alpha_i = 1$ when $\xi_i > 0$ (which is when $y_i f(x_i) < 1$). Also when $y_i f(x_i) > 1$, $\xi_i = 0$ since no cost is incurred, and $\alpha_i = 0$. When $y_i f(x_i) = 1$, α_i can lie between 0 and 1.[2]

We wish to find the entire solution path for all values of $\lambda \ge 0$. The basic idea of our algorithm is as follows. We start with λ large and decrease it toward zero, keeping track of all the events that occur along the way. As λ decreases, $||\beta||$ increases, and hence the width of the margin decreases (see Figure 1). As this width decreases, points move from being inside to outside the margin. Their corresponding α_i change from $\alpha_i = 1$ when they are inside the margin ($y_i f(x_i) < 1$) to $\alpha_i = 0$ when they are outside the margin ($y_i f(x_i) > 1$). By continuity, points must linger on the margin ($y_i f(x_i) = 1$) while their α_i decrease from 1 to 0. We will see that the $\alpha_i(\lambda)$ trajectories are piecewise-linear in λ, which affords a great computational savings: as long as we can establish the break points, all

2. For readers more familiar with the traditional SVM formulation (4), we note that there is a simple connection between the corresponding Lagrange multipliers, $\alpha'_i = \alpha_i/\lambda = C\alpha_i$, and hence in that case $\alpha'_i \in [0, C]$. We prefer our formulation here since our $\alpha_i \in [0,1]$, and this simplifies the definition of the paths we define.

values in between can be found by simple linear interpolation. Note that points can return to the margin, after having passed through it.

It is easy to show that if the $\alpha_i(\lambda)$ are piecewise linear in λ, then both $\alpha_i'(C) = C\alpha_i(C)$ and $\beta(C)$ are piecewise linear in C. It turns out that $\beta_0(C)$ is also piecewise linear in C. We will frequently switch between these two representations.

We denote by I_+ the set of indices corresponding to $y_i = +1$ points, there being $n_+ = |I_+|$ in total. Likewise for I_- and n_-. Our algorithm keeps track of the following sets (with names inspired by the hinge loss function in Figure 2):

- $\mathcal{E} = \{i : y_i f(x_i) = 1,\ 0 \le \alpha_i \le 1\}$, $\mathcal{E}$ for Elbow,
- $\mathcal{L} = \{i : y_i f(x_i) < 1,\ \alpha_i = 1\}$, $\mathcal{L}$ for Left of the elbow,
- $\mathcal{R} = \{i : y_i f(x_i) > 1,\ \alpha_i = 0\}$, $\mathcal{R}$ for Right of the elbow.

3. Initialization

We need to establish the initial state of the sets defined above. When λ is very large (∞), from (10) $\beta = 0$, and the initial values of β_0 and the α_i depend on whether $n_- = n_+$ or not. If the classes are balanced, one can directly find the initial configuration by finding the most extreme points in each class. We will see that when $n_- \neq n_+$, this is no longer the case, and in order to satisfy the constraint (11), a quadratic programming algorithm is needed to obtain the initial configuration.

In fact, our `SvmPath` algorithm can be started at any intermediate solution of the SVM optimization problem (i.e. the solution for any λ), since the values of α_i and $f(x_i)$ determine the sets $\mathcal{L}$, $\mathcal{E}$ and $\mathcal{R}$. We will see in Section 6 that if there is no intercept in the model, the initialization is again trivial, no matter whether the classes are balanced or not. We have prepared some MPEG movies to illustrate the two special cases detailed below. The movies can be downloaded at the web site `http://www-stat.stanford.edu/~hastie/Papers/svm/MOVIE/`.

3.1 Initialization: $n_- = n_+$

Lemma 1 *For λ sufficiently large, all the $\alpha_i = 1$. The initial $\beta_0 \in [-1,1]$ — any value gives the same loss $\sum_{i=1}^n \xi_i = n_+ + n_-$.*

Proof Our proof relies on the criterion and the KKT conditions in Section 2. Since $\beta = 0$, $f(x) = \beta_0$. To minimize $\sum_{i=1}^n \xi_i$, we should clearly restrict β_0 to $[-1,1]$. For $\beta_0 \in (-1,1)$, all the $\xi_i > 0$, $\gamma_i = 0$ in (12), and hence $\alpha_i = 1$. Picking one of the endpoints, say $\beta_0 = -1$, causes $\alpha_i = 1,\ i \in I_+$, and hence also $\alpha_i = 1,\ i \in I_-$, for (11) to hold. ■

We also have that for these early and large values of λ

$$\beta = \frac{1}{\lambda}\beta^* \text{ where } \beta^* = \sum_{i=1}^n y_i x_i. \tag{15}$$

Now in order that (11) remain satisfied, we need that one or more positive and negative examples hit the elbow *simultaneously*. Hence as λ decreases, we require that $\forall i\ y_i f(x_i) \le 1$ or

$$y_i \left[\frac{\beta^{*T} x_i}{\lambda} + \beta_0 \right] \le 1 \tag{16}$$

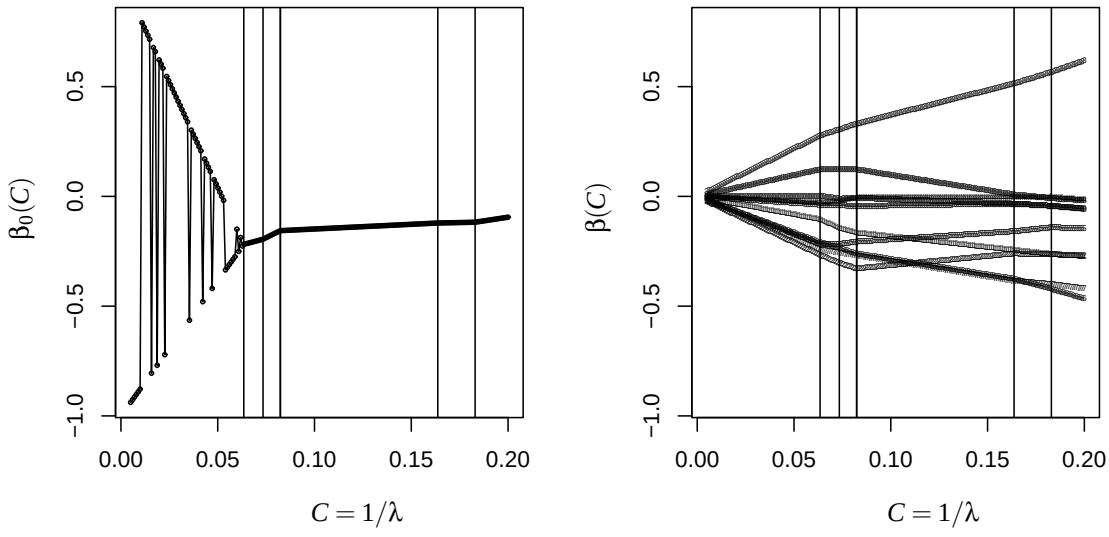


Figure 5: The initial paths of the coefficients in a small simulated dataset with $n_- = n_+$. We see the zone of allowable values for β_0 shrinking toward a fixed point (20). The vertical lines indicate the breakpoints in the piecewise linear coefficient paths.

or

$$\beta_0 \leq 1 - \frac{\beta^{*T} x_i}{\lambda} \quad \text{for all } i \in I_+ \tag{17}$$

$$\beta_0 \geq -1 - \frac{\beta^{*T} x_i}{\lambda} \quad \text{for all } i \in I_-. \tag{18}$$

Pick $i_+ = \arg\max_{i \in I_+} \beta^{*T} x_i$ and $i_- = \arg\min_{i \in I_-} \beta^{*T} x_i$ (for simplicity we assume that these are unique). Then at this point of entry and beyond for a while we have $\alpha_{i_+} = \alpha_{i_-}$, and $f(x_{i_+}) = 1$ and $f(x_{i_-}) = -1$. This gives us two equations to solve for the initial point of entry λ_0 and β_0, with solutions

$$\lambda_0 = \frac{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}}{2}, \tag{19}$$

$$\beta_0 = -\left(\frac{\beta^{*T} x_{i_+} + \beta^{*T} x_{i_-}}{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}} \right). \tag{20}$$

Figure 5 (left panel) shows a trajectory of $\beta_0(C)$ as a function of C, for a small simulated data set. These solutions were computed directly using a quadratic-programming algorithm, using a

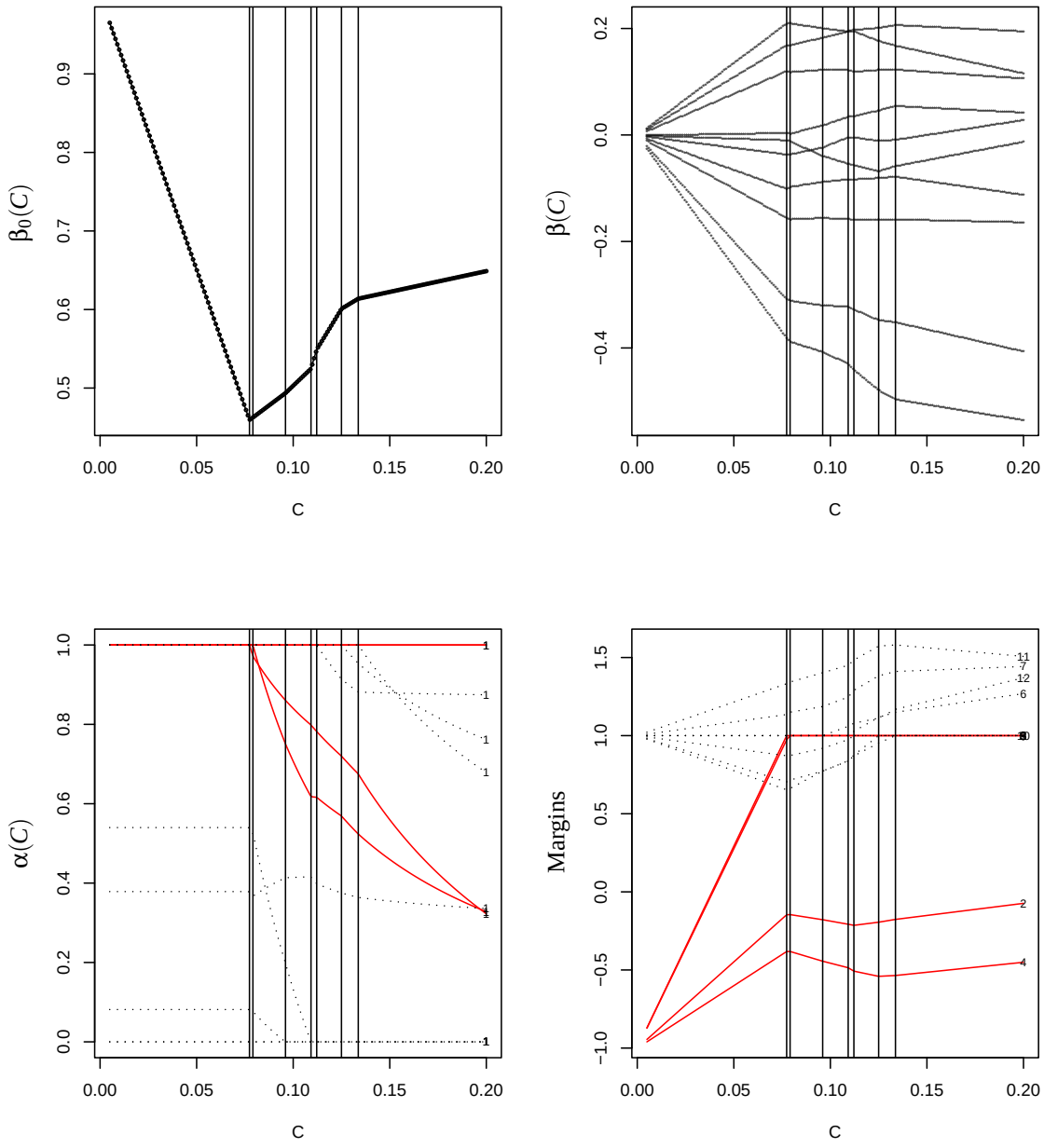


Figure 6: The initial paths of the coefficients in a case where $n_- < n_+$. All the n_- points are misclassified, and start off with a margin of -1. The α_i^* remain constant until one of the points in I_- reaches the margin. The vertical lines indicate the breakpoints in the piecewise linear $\beta(C)$ paths. Note that the $\alpha_i(C)$ are *not* piecewise linear in C, but rather in $\lambda = 1/C$.

predefined grid of values for λ. The arbitrariness of the initial values is indicated by the zig-zag nature of this path. The breakpoints were found using our exact-path algorithm.

3.2 Initialization: $n_+ > n_-$

In this case, when $\beta = 0$, the optimal choice for β_0 is 1, and the loss is $\sum_{i=1}^n \xi_i = n_-$. However, we also require that (11) holds.

Lemma 2 *With* $\beta^*(\alpha) = \sum_{i=1}^n y_i \alpha_i x_i$, *let*

$$\{\alpha_i^*\} = \arg\min_{\alpha} ||\beta^*(\alpha)||^2 \quad (21)$$

$$\text{s.t. } \alpha_i \in [0,1] \text{ for } i \in I_+,\ \alpha_i = 1 \text{ for } i \in I_-, \text{ and } \textstyle\sum_{i \in I_+} \alpha_i = n_- \quad (22)$$

Then for some λ_0 *we have that for all* $\lambda > \lambda_0$, $\alpha_i = \alpha_i^*$, *and* $\beta = \beta^*/\lambda$, *with* $\beta^* = \sum_{i=1}^n y_i \alpha_i^* x_i$.

Proof The Lagrange dual corresponding to (9) is obtained by substituting (10)–(12) into (9) (Hastie et al., 2001, Equation 12.13):

$$L_D = \sum_{i=1}^n \alpha_i - \frac{1}{2\lambda} \sum_{i=1}^n \sum_{i'=1}^n \alpha_i \alpha_{i'} y_i y_{i'} x_i x_{i'}. \quad (23)$$

Since we start with $\beta = 0$, $\beta_0 = 1$, all the I_- points are misclassified, and hence we will have $\alpha_i = 1 \forall i \in I_-$, and hence from (11) $\sum_{i=1}^n \alpha_i = 2n_-$. This latter sum will remain $2n_-$ for a while as β grows away from zero. This means that during this phase, the first term in the Lagrange dual is constant; the second term is equal to $-\frac{1}{2\lambda}||\beta^*(\alpha)||^2$, and since we maximize the dual, this proves the result. ■

We now establish the "starting point" λ_0 and β_0 when the α_i start to change. Let β^* be the fixed coefficient direction corresponding to α_i^* (as in (15)):

$$\beta^* = \sum_{i=1}^n \alpha_i^* y_i x_i. \quad (24)$$

There are two possible scenarios:

1. There exist two or more elements in I_+ with $0 < \alpha_i^* < 1$, or
2. $\alpha_i^* \in \{0,1\}\ \forall i \in I_+$.

Consider the first scenario (depicted in Figure 6), and suppose $\alpha_{i_+}^* \in (0,1)$ (on the margin). Let $i_- = \arg\min_{i \in I_-} \beta^{*T} x_i$. Then since the point i_+ remains on the margin until an I_- point reaches its margin, we can find

$$\lambda_0 = \frac{\beta^{*T} x_{i_+} - \beta^{*T} x_{i_-}}{2}, \quad (25)$$

identical in form to to (19), as is the corresponding β_0 to (20).

For the second scenario, it is easy to see that we find ourselves in the same situation as in Section 3.1—a point from I_- and one of the points in I_+ with $\alpha_i^* = 1$ must reach the margin simultaneously. Hence we get an analogous situation, except with $i_+ = \arg\max_{i \in I_+^1} \beta^{*T} x_i$, where I_+^1 is the subset of I_+ with $\alpha_i^* = 1$.

3.3 Kernels

The development so far has been in the original feature space, since it is easier to visualize. It is easy to see that the entire development carries through with "kernels" as well. In this case $f(x) = \beta_0 + g(x)$, and the only change that occurs is that (10) is changed to

$$g(x_i) = \frac{1}{\lambda}\sum_{j=1}^{n}\alpha_j y_j K(x_i, x_j),\ i = 1,\ldots,n, \tag{26}$$

or $\theta_j(\lambda) = \alpha_j y_j/\lambda$ using the notation in (7).

Our initial conditions are defined in terms of expressions $\beta^{*T} x_{i_+}$, for example, and again it is easy to see that the relevant quantities are

$$g^*(x_{i^+}) = \sum_{j=1}^{n}\alpha_j^* y_j K(x_{i_+}, x_j), \tag{27}$$

where the α_i^* are all 1 in Section 3.1, and defined by Lemma 2 in Section 3.2.

Hereafter we will develop our algorithm for this more general kernel case.

4. The Path

The algorithm hinges on the set of points $\mathcal{E}$ sitting at the elbow of the loss function — i.e on the margin. These points have $y_i f(x_i) = 1$ and $\alpha_i \in [0,1]$. These are distinct from the points $\mathcal{R}$ to the right of the elbow, with $y_i f(x_i) > 1$ and $\alpha_i = 0$, and those points $\mathcal{L}$ to the left with $y_i f(x_i) < 1$ and $\alpha_i = 1$. We consider this set at the point that an event has occurred. The event can be either:

1. The initial event, which means 2 or more points start at the elbow, with their initial values of $\alpha \in [0,1]$.

2. A point from $\mathcal{L}$ has just entered $\mathcal{E}$, with its value of α_i initially 1.

3. A point from $\mathcal{R}$ has reentered $\mathcal{E}$, with its value of α_i initially 0.

4. One or more points in $\mathcal{E}$ has left the set, to join either $\mathcal{R}$ or $\mathcal{L}$.

Whichever the case, for continuity reasons this set will stay stable until the next event occurs, since to pass through $\mathcal{E}$, a point's α_i must change from 0 to 1 or vice versa. Since all points in $\mathcal{E}$ have $y_i f(x_i) = 1$, we can establish a path for their α_i.

Event 4 allows for the possibility that $\mathcal{E}$ becomes empty while $\mathcal{L}$ is not. If this occurs, then the KKT condition (11) implies that $\mathcal{L}$ is balanced w.r.t. +1s and -1s, and we resort to the initial condition as in Section 3.1.

We use the subscript ℓ to index the sets above immediately after the ℓth event has occurred. Suppose $|\mathcal{E}_\ell| = m$, and let α_i^ℓ, β_0^ℓ and λ_ℓ be the values of these parameters at the point of entry. Likewise f^ℓ is the function at this point. For convenience we define $\alpha_0 = \lambda\beta_0$, and hence $\alpha_0^\ell = \lambda_\ell\beta_0^\ell$.

Since

$$f(x) = \frac{1}{\lambda}\left(\sum_{j=1}^{n} y_j\alpha_j K(x, x_j) + \alpha_0\right), \tag{28}$$

for $\lambda_\ell > \lambda > \lambda_{\ell+1}$ we can write

$$\begin{aligned} f(x) &= \left[f(x) - \frac{\lambda_\ell}{\lambda} f^\ell(x)\right] + \frac{\lambda_\ell}{\lambda} f^\ell(x) \\ &= \frac{1}{\lambda}\left[\sum_{j\in\mathcal{E}_\ell} (\alpha_j - \alpha_j^\ell) y_j K(x, x_j) + (\alpha_0 - \alpha_0^\ell) + \lambda_\ell f^\ell(x)\right]. \end{aligned} \tag{29}$$

The second line follows because all the observations in $\mathcal{L}_\ell$ have their $\alpha_i = 1$, and those in $\mathcal{R}_\ell$ have their $\alpha_i = 0$, for this range of λ. Since each of the m points $x_i \in \mathcal{E}_\ell$ are to stay at the elbow, we have that

$$\frac{1}{\lambda}\left[\sum_{j\in\mathcal{E}_\ell} (\alpha_j - \alpha_j^\ell) y_i y_j K(x_i, x_j) + y_i(\alpha_0 - \alpha_0^\ell) + \lambda_\ell\right] = 1,\ \forall i \in \mathcal{E}_\ell. \tag{30}$$

Writing $\delta_j = \alpha_j^\ell - \alpha_j$, from (30) we have

$$\sum_{j\in\mathcal{E}_\ell} \delta_j y_i y_j K(x_i, x_j) + y_i \delta_0 = \lambda_\ell - \lambda,\ \forall i \in \mathcal{E}_\ell. \tag{31}$$

Furthermore, since at all times $\sum_{i=1}^n y_i \alpha_i = 0$, we have that

$$\sum_{j\in\mathcal{E}_\ell} y_j \delta_j = 0. \tag{32}$$

Equations (31) and (32) constitute $m+1$ linear equations in $m+1$ unknowns δ_j, and can be solved.

Denoting by $\mathbf{K}_\ell^*$ the $m \times m$ matrix with ijth entry $y_i y_j K(x_i, x_j)$ for i and j in $\mathcal{E}_\ell$, we have from (31) that

$$\mathbf{K}_\ell^* \delta + \delta_0 \mathbf{y}_\ell = (\lambda_\ell - \lambda)\mathbf{1}, \tag{33}$$

where $\mathbf{y}_\ell$ is the m vector with entries $y_i,\ i \in \mathcal{E}_\ell$. From (32) we have

$$\mathbf{y}_\ell^T \delta = 0. \tag{34}$$

We can combine these two into one matrix equation as follows. Let

$$\mathbf{A}_\ell = \begin{pmatrix} 0 & \mathbf{y}_\ell{}^T \\ \mathbf{y}_\ell & \mathbf{K}_\ell^* \end{pmatrix}, \quad \delta^a = \begin{pmatrix} \delta_0 \\ \delta \end{pmatrix}, \text{ and } \quad \mathbf{1}^a = \begin{pmatrix} 0 \\ \mathbf{1} \end{pmatrix}, \tag{35}$$

then (34) and (33) can be written

$$\mathbf{A}_\ell \delta^a = (\lambda_\ell - \lambda)\mathbf{1}^a. \tag{36}$$

If $\mathbf{A}_\ell$ has full rank, then we can write

$$\mathbf{b}^a = \mathbf{A}_\ell{}^{-1}\mathbf{1}^a, \tag{37}$$

and hence

$$\alpha_j = \alpha_j^\ell - (\lambda_\ell - \lambda) b_j,\ j \in \{0\} \cup \mathcal{E}_\ell. \tag{38}$$

Hence for $\lambda_{\ell+1} < \lambda < \lambda_\ell$, the α_j for points at the elbow proceed *linearly in* λ. From (29) we have

$$f(x) = \frac{\lambda_\ell}{\lambda}\left[f^\ell(x) - h^\ell(x)\right] + h^\ell(x), \tag{39}$$

where

$$h^{\ell}(x) = \sum_{j \in \mathcal{E}_{\ell}} y_j b_j K(x, x_j) + b_0. \tag{40}$$

Thus the function itself changes in a piecewise-inverse manner in λ.

If $\mathbf{A}_{\ell}$ does not have full rank, then the solution paths for some of the α_i are not unique, and more care has to be taken in solving the system (36). This occurs, for example, when two training observations are identical (tied in x and y). Other degeneracies can occur, but rarely in practice, such as three different points on the same margin in $\mathbb{R}^2$. These issues and some of the related updating and downdating schemes are an area we are currently researching, and will be reported elsewhere.

4.1 Finding $\lambda_{\ell+1}$

The paths (38)–(39) continue until one of the following events occur:

1. One of the α_i for $i \in \mathcal{E}_{\ell}$ reaches a boundary (0 or 1). For each i the value of λ for which this occurs is easily established from (38).

2. One of the points in $\mathcal{L}^{\ell}$ or $\mathcal{R}^{\ell}$ attains $y_i f(x_i) = 1$. From (39) this occurs for point i at

$$\lambda = \lambda_{\ell} \left(\frac{f^{\ell}(x_i) - h^{\ell}(x_i)}{y_i - h^{\ell}(x_i)} \right). \tag{41}$$

By examining these conditions, we can establish the largest $\lambda < \lambda_{\ell}$ for which an event occurs, and hence establish $\lambda_{\ell+1}$ and update the sets.

One special case not addressed above is when the set $\mathcal{E}$ becomes empty during the course of the algorithm. In this case, we revert to an initialization setup using the points in $\mathcal{L}$. It must be the case that these points have an equal number of +1's as -1's, and so we are in the balanced situation as in 3.1.

By examining in detail the linear boundary in examples where $p = 2$, we observed several different types of behavior:

1. If $|\mathcal{E}| = 0$, than as λ decreases, the orientation of the decision boundary stays fixed, but the margin width narrows as λ decreases.

2. If $|\mathcal{E}| = 1$ or $|\mathcal{E}| = 2$, but with the pair of points of opposite classes, then the orientation typically rotates as the margin width gets narrower.

3. If $|\mathcal{E}| = 2$, with both points having the same class, then the orientation remains fixed, with the one margin stuck on the two points as the decision boundary gets shrunk toward it.

4. If $|\mathcal{E}| \geq 3$, then the margins and hence $f(x)$ remains fixed, as the $\alpha_i(\lambda)$ change. This implies that $h^{\ell} = f^{\ell}$ in (39).

4.2 Termination

In the separable case, we terminate when $\mathcal{L}$ becomes empty. At this point, all the ξ_i in (8) are zero, and further movement increases the norm of β unnecessarily.

In the non-separable case, λ runs all the way down to zero. For this to happen without f "blowing up" in (39), we must have $f^{\ell} - h^{\ell} = 0$, and hence the boundary and margins remain

fixed at a point where $\sum_i \xi_i$ is as small as possible, and the margin is as wide as possible subject to this constraint.

4.3 Computational Complexity

At any update event ℓ along the path of our algorithm, the main computational burden is solving the system of equations of size $m_\ell = |\mathcal{E}_\ell|$. While this normally involves $O(m_\ell^3)$ computations, since $\mathcal{E}_{\ell+1}$ differs from $\mathcal{E}_\ell$ by typically one observation, inverse updating/downdating can reduce the computations to $O(m_\ell^2)$. The computation of $h^\ell(x_i)$ in (40) requires $O(nm_\ell)$ computations. Beyond that, several checks of cost $O(n)$ are needed to evaluate the next move.

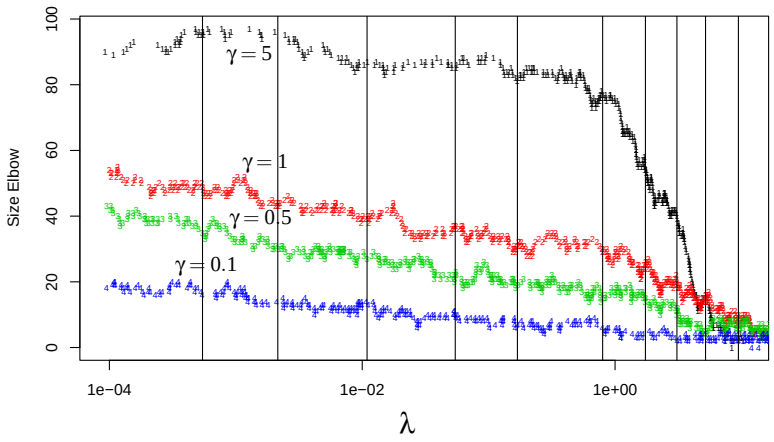


Figure 7: The elbow sizes $|\mathcal{E}_\ell|$ as a function of λ, for different values of the radial-kernel parameter γ. The vertical lines show the positions used to compare the times with `libsvm`.

We have explored using partitioned inverses for updating/downdating the solutions to the elbow equations (for the nonsingular case), and our experiences are mixed. In our R implementations, the computational savings appear negligible for the problems we have tackled, and after repeated updating, rounding errors can cause drift. At the time of this publication, we in fact do not use updating at all, and simply solve the system each time. We are currently exploring numerically stable ways for managing these updates.

Although we have no hard results, our experience so far suggests that the total number Λ of moves is $O(k\min(n_+, n_-))$, for k around $4-6$; hence typically some small multiple c of n. If the average size of $\mathcal{E}_\ell$ is m, this suggests the total computational burden is $O(cn^2m + nm^2)$, which is similar to that of a single SVM fit.

Our R function `SvmPath` computes all 632 steps in the mixture example ($n_+ = n_- = 100$, radial kernel, $\gamma = 1$) in 1.44(0.02) secs on a Pentium 4, 2Ghz Linux machine; the `svm` function (using the optimized code `libsvm`, from the R library `e1071`) takes 9.28(0.06) seconds to compute the solution at 10 points along the path. Hence it takes our procedure about 50% more time to compute the entire path, than it costs `libsvm` to compute a typical single solution.

We often wish to make predictions at new inputs. We can also do this efficiently for all values of λ, because from (28) we see that (modulo $1/\lambda$), these also change in a piecewise-linear fashion in λ. Hence we can compute the entire fit path for a single input x in $O(n)$ calculations, plus an additional $O(nq)$ operations to compute the kernel evaluations (assuming it costs $O(q)$ operations to compute $K(x, x_i)$).

5. Examples

In this section we look at three examples, two synthetic and one real. We examine our running mixture example in some more detail, and expose the nature of quadratic regularization in the kernel feature space. We then simulate and examine a scaled-down version of the $p \gg n$ problem—many more inputs than samples. Despite the fact that perfect separation is possible with large margins, a heavily regularized model is optimal in this case. Finally we fit SVM path models to some microarray cancer data.

5.1 Mixture Simulation

In Figure 4 we show the test-error curves for a large number of values of λ, and four different values for γ for the radial kernel. These λ_ℓ are in fact the *entire* collection of change points as described in Section 4. For example, for the second panel, with $\gamma = 1$, there are 623 change points. Figure 8 [upper plot] shows the paths of all the $\alpha_i(\lambda)$, as well as [lower plot] a few individual examples. An MPEG movie of the sequence of models can be downloaded from the first author's website.

We were at first surprised to discover that not all these sequences achieved zero training errors on the 200 training data points, at their least regularized fit. In fact the minimal training errors, and the corresponding values for γ are summarized in Table 1. It is sometimes argued that the implicit

γ	5	1	0.5	0.1
Training Errors	0	12	21	33
Effective Rank	200	177	143	76

Table 1: The number of minimal training errors for different values of the radial kernel scale parameter γ, for the mixture simulation example. Also shown is the effective rank of the 200×200 Gram matrix $\mathbf{K}_\gamma$.

feature space is "infinite dimensional" for this kernel, which suggests that perfect separation is always possible. The last row of the table shows the effective rank of the kernel *Gram* matrix $\mathbf{K}$ (which we defined to be the number of singular values greater than 10^{-12}). This 200×200 matrix has elements $\mathbf{K}_{i,j} = K(x_i, x_j)$, $i, j = 1, \ldots, n$. In general a full rank $\mathbf{K}$ is required to achieve perfect separation. Similar observations have appeared in the literature (Bach and Jordan, 2002; Williams and Seeger, 2000).

This emphasizes the fact that not all features in the feature map implied by K are of equal stature; many of them are shrunk way down to zero. Alternatively, the regularization in (6) and (7) penalizes unit-norm features by the inverse of their eigenvalues, which effectively annihilates some, depending on γ. Small γ implies wide, flat kernels, and a suppression of wiggly, "rough" functions.

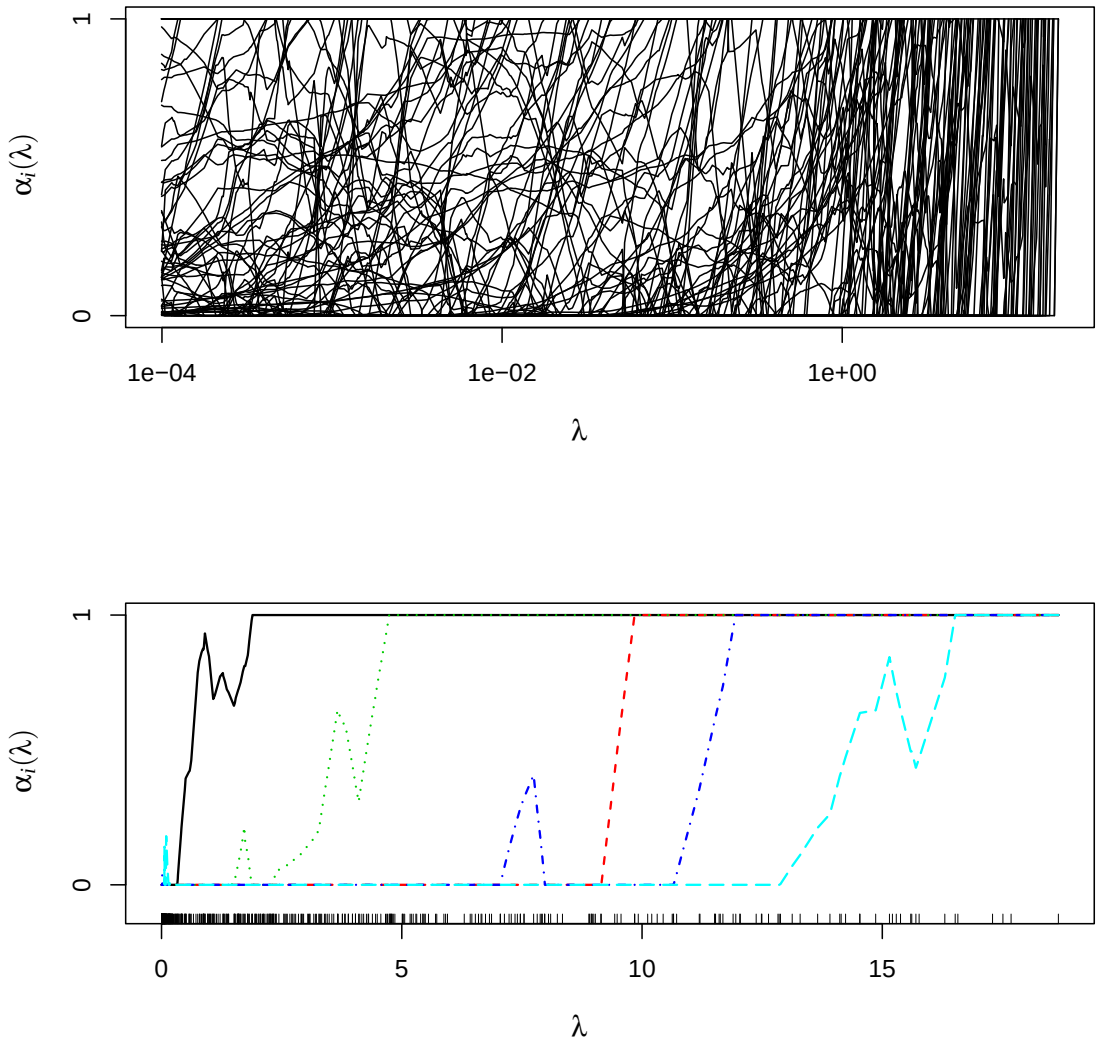


Figure 8: [Upper plot] The entire collection of piece-wise linear paths $\alpha_i(\lambda)$, $i = 1,\ldots,N$, for the mixture example. Note: λ is plotted on the log-scale. [Lower plot] Paths for 5 selected observations; λ is *not* on the log scale.

Writing (7) in matrix form,

$$\min_{\beta_0,\theta} L[\mathbf{y}, \mathbf{K}\theta] + \frac{\lambda}{2}\theta^T\mathbf{K}\theta, \tag{42}$$

we reparametrize using the eigen-decomposition of $\mathbf{K} = \mathbf{U}\mathbf{D}\mathbf{U}^T$. Let $\mathbf{K}\theta = \mathbf{U}\theta^*$ where $\theta^* = \mathbf{D}\mathbf{U}^T\theta$. Then (42) becomes

$$\min_{\beta_0,\theta^*} L[\mathbf{y}, \mathbf{U}\theta^*] + \frac{\lambda}{2}\theta^{*T}\mathbf{D}^{-1}\theta^*. \tag{43}$$

Now the columns of $\mathbf{U}$ are unit-norm basis functions (in $\mathbb{R}^2$) spanning the column space of $\mathbf{K}$;

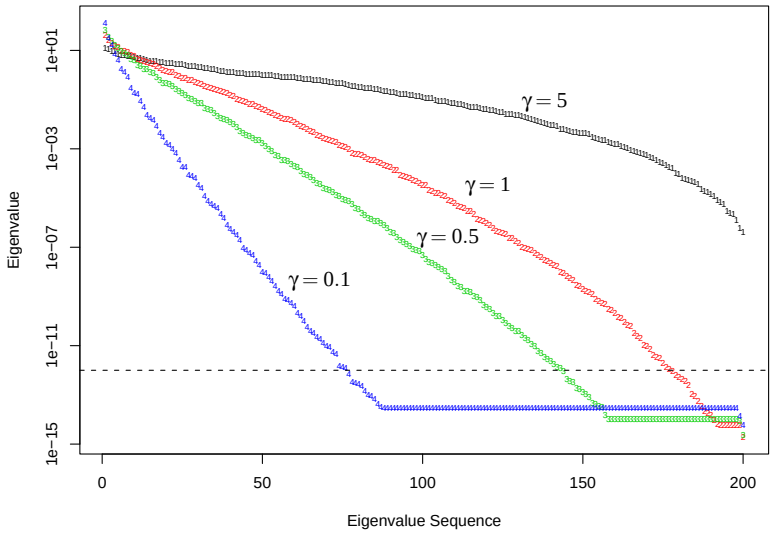


Figure 9: The eigenvalues (on the log scale) for the kernel matrices $\mathbf{K}_\gamma$ corresponding to the four values of γ as in Figure 4. The larger eigenvalues correspond in this case to smoother eigenfunctions, the small ones to rougher. The rougher eigenfunctions get penalized exponentially more than the smoother ones. For smaller values of γ, the effective dimension of the space is truncated.

from (43) we see that those members corresponding to near-zero eigenvalues (the elements of the diagonal matrix $\mathbf{D}$) get heavily penalized and hence ignored. Figure 9 shows the elements of $\mathbf{D}$ for the four values of γ. See Hastie et al. (2001, Chapter 5) for more details.

5.2 p≫n Simulation

The SVM is popular in situations where the number of features exceeds the number of observations. Gene expression arrays are a leading example, where a typical dataset has $p > 10,000$ while $n < 100$. Here one typically fits a linear classifier, and since it is easy to separate the data, the optimal marginal classifier is the *de facto* choice. We argue here that regularization can play an important role for these kinds of data.

We mimic a simulation found in Marron (2003). We have $p = 50$ and $n = 40$, with a 20-20 split of "+" and "-" class members. The x_{ij} are all iid realizations from a $N(0,1)$ distribution, except for the first coordinate, which has mean +2 and -2 in the respective classes.[3] The Bayes classifier in this case uses only the first coordinate of x, with a threshold at 0. The Bayes risk is 0.012. Figure 10 summarizes the experiment. We see that the most regularized models do the best here, not the maximal margin classifier.

Although the most regularized linear SVM is the best in this example, we notice a disturbing aspect of its endpoint behavior in the top-right plot. Although β is determined by all the points, the threshold β_0 is determined by the two most extreme points in the two classes (see Section 3.1). This can lead to irregular behavior, and indeed in some realizations from this model this was the case. For values of λ larger than the initial value λ_1, we saw in Section 3 that the endpoint behavior depends on whether the classes are balanced or not. In either case, as λ increases, the error converges to the estimated null error rate n_{min}/n.

This same objection is often made at the other extreme of the optimal margin; however, it typically involves more support points (19 points on the margin here), and tends to be more stable (but still no good in this case). For solutions in the interior of the regularization path, these objections no longer hold. Here the regularization forces more points to overlap the margin (support points), and hence determine its orientation.

Included in the figures are regularized linear discriminant analysis and logistic regression models (using the same λ_ℓ sequence as the SVM). Both show similar behavior to the regularized SVM, having the most regularized solutions perform the best. Logistic regression can be seen to assign weights $p_i(1-p_i)$ to observations in the fitting of its coefficients β and β_0, where

$$p_i = \frac{1}{1+e^{-\beta_0-\beta^T x_i}} \tag{44}$$

is the estimated probability of $+1$ occurring at x_i (Hastie and Tibshirani, 1990, e.g.).

- Since the decision boundary corresponds to $p(x) = 0.5$, these weights can be seen to die down in a quadratic fashion from $1/4$, as we move away from the boundary.
- The rate at which the weights die down with distance from the boundary depends on $||\beta||$; the smaller this norm, the slower the rate.

It can be shown, for separated classes, that the limiting solution ($\lambda \downarrow 0$) for the regularized logistic regression model is identical to the SVM solution: the maximal margin separator (Rosset et al., 2003).

Not surprisingly, given the similarities in their loss functions (Figure 2), both regularized SVMs and logistic regression involve more or less observations in determining their solutions, depending on the amount of regularization. This "involvement" is achieved in a smoother fashion by logistic regression.

5.3 Microarray Classification

We illustrate our algorithm on a large cancer expression data set (Ramaswamy et al., 2001). There are 144 training tumor samples and 54 test tumor samples, spanning 14 common tumor classes that

3. Here we have one important feature; the remaining 49 are noise. With expression arrays, the important features typically occur in groups, but the total number p is much larger.

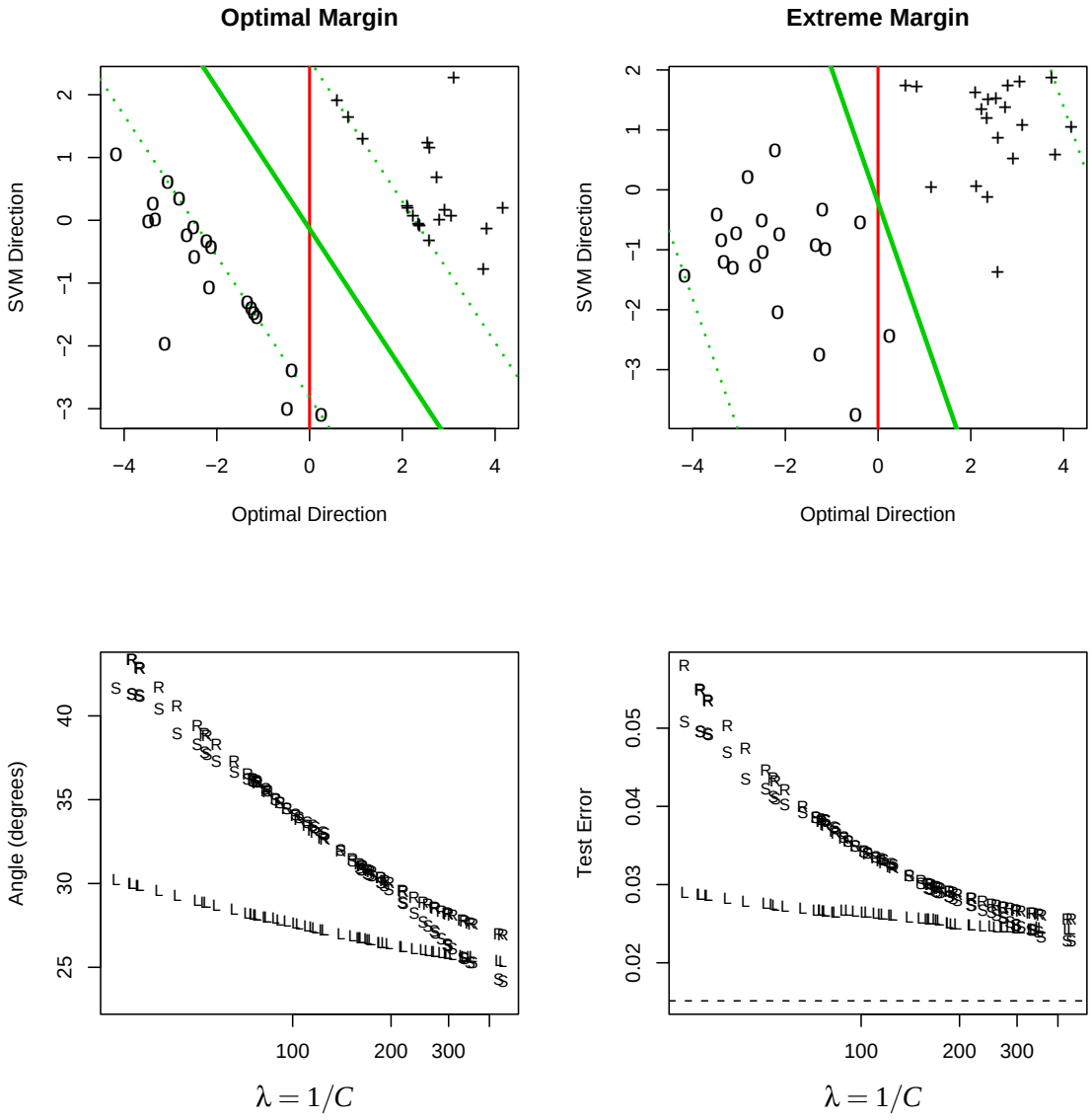


Figure 10: $p \gg n$ simulation. [Top Left] The training data projected onto the space spanned by the (known) optimal coordinate 1, and the optimal margin coefficient vector found by a non-regularized SVM. We see the large gap in the margin, while the Bayes-optimal classifier (vertical red line) is actually *expected* to make a small number of errors. [Top Right] The same as the left panel, except we now project onto the most regularized SVM coefficient vector. This solution is closer to the Bayes-optimal solution. [Lower Left] The angles between the Bayes-optimal direction, and the directions found by the SVM (S) along the regularized path. Included in the figure are the corresponding coefficients for regularized LDA (R)(Hastie et al., 2001, Chapter 4) and regularized logistic regression (L)(Zhu and Hastie, 2004), using the same quadratic penalties. [Lower Right] The test errors corresponding to the three paths. The horizontal line is the estimated Bayes rule using only the first coordinate.

account for 80% of new cancer diagnoses in the U.S.A. There are 16,063 genes for each sample. Hence $p = 16,063$ and $n = 144$. We denote the number of classes by $K = 14$. A goal is to build a classifier for predicting the cancer class of a new sample, given its expression values.

We used a common approach for extending the SVM from two-class to multi-class classification:

1. Fit K different SVM models, each one classifying a single cancer class (+1) versus the rest (-1).
2. Let $[f_1^\lambda(x), \ldots, f_K^\lambda(x)]$ be the vector of evaluations of the fitted functions (with parameter λ) at a test observation x.
3. Classify $C^\lambda(x) = \arg\max_k f_k^\lambda(x)$.

Other, more direct, multi-class generalizations exist (Rosset et al., 2003; Weston and Watkins, 1998); although exact path algorithms are possible here too, we were able to implement our approach most easily with the "one vs all" strategy above. Figure 11 shows the results of fitting this family of SVM models. Shown are the training error, test error, as well as 8-fold balanced cross-validation.[4] The training error is zero everywhere, but both the test and CV error increase sharply when the model is too regularized. The right plot shows similar results using quadratically regularized multinomial regression (Zhu and Hastie, 2004).

Although the least regularized SVM and multinomial models do the best, this is still not very good. With fourteen classes, this is a tough classification problem.

It is worth noting that:

- The 14 different classification problems are very "lop-sided"; in many cases 8 observations in one class vs the 136 others. This tends to produce solutions with all members of the small class on the boundary, a somewhat unnatural situation.
- For both the SVM and the quadratically regularized multinomial regression, one can reduce the logistics by pre-transforming the data. If $\mathbf{X}$ is the $n \times p$ data matrix, with $p \gg n$, let its singular-value decomposition be $\mathbf{UDV}^T$. We can replace $\mathbf{X}$ by the $n \times n$ matrix $\mathbf{XV} = \mathbf{UD} = \mathbf{R}$ and obtain identical results (Hastie and Tibshirani, 2003). The same transformation $\mathbf{V}$ is applied to the test data. This transformation is applied once upfront, and the transformed data is used in all subsequent analyses (i.e. K-fold cross-validation as well).

6. No Intercept and Kernel Density Classification

Here we consider a simplification of the models (6) and (7) where we leave out the intercept term β_0. It is easy to show that the solution for $g(x)$ has the identical form as in (26):

$$g(x) = \frac{1}{\lambda} \sum_{j=1}^{n} \alpha_j y_j K(x, x_j). \tag{45}$$

However, $f(x) = g(x)$ (or $f(x) = \beta^T x$ in the linear case), and we lose the constraint (11) due to the intercept term.

This also adds considerable simplification to our algorithm, in particular the initial conditions.

4. By balanced we mean the 14 cancer classes were represented equally in each of the folds; 8 folds were used to accommodate this balance, since the class sizes in the training set were multiples of 8.

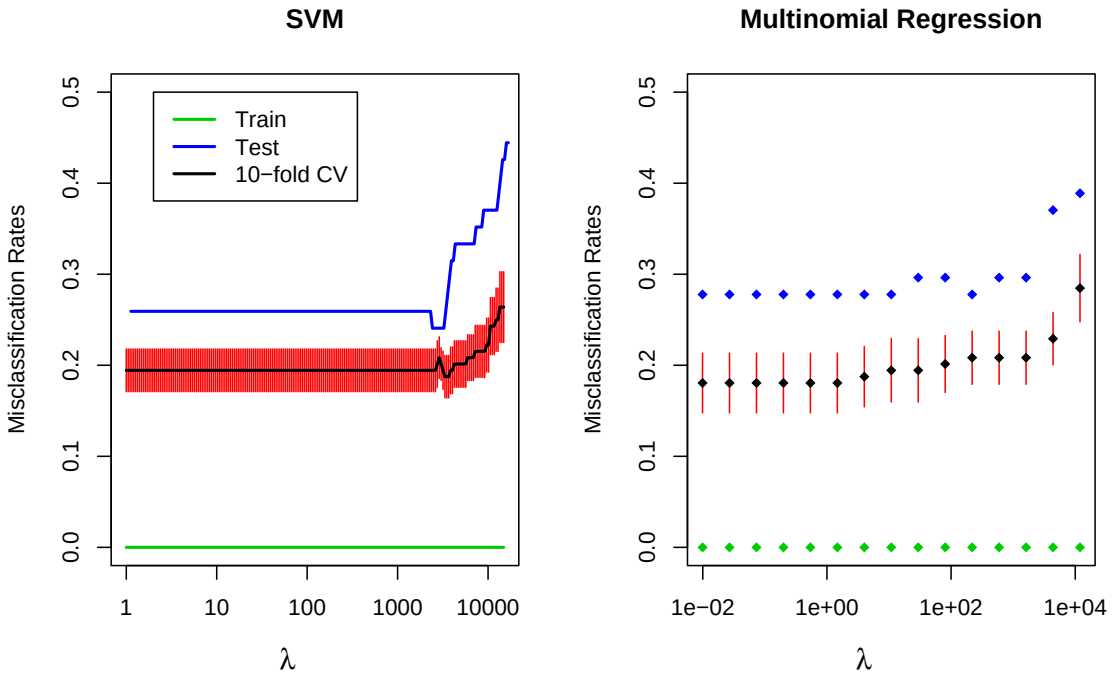


Figure 11: Misclassification rates for cancer classification by gene expression measurements. The left panel shows the the training (lower green), cross-validation (middle black, with standard errors) and test error (upper blue) curves for the entire SVM path. Although the CV and test error curves appear to have quite different levels, the region of interesting behavior is the same (with a curious dip at about $\lambda = 3000$). Seeing the entire path leaves no guesswork as to where the region of interest might be. The right panel shows the same for the regularized multiple logistic regression model. Here we do not have an exact path algorithm, so a grid of 15 values of λ is used (on a log scale).

- It is easy to see that initially $\alpha_i = 1 \forall i$, since $f(x)$ is close to zero for large λ, and hence all points are in $\mathcal{L}$. This is true whether or not $n_- = n_+$, unlike the situation when an intercept is present (Section 3.2).

- With $f^*(x) = \sum_{j=1}^n y_j K(x, x_j)$, the first element of $\mathcal{E}$ is $i^* = \arg\max_i |f^*(x_i)|$, with $\lambda_1 = |f^*(x_{i^*})|$. For $\lambda \in [\lambda_1, \infty)$, $f(x) = f^*(x)/\lambda$.

- The linear equations that govern the points in $\mathcal{E}$ are similar to (33):

$$\mathbf{K}_\ell^* \delta = (\lambda_\ell - \lambda)\mathbf{1}, \tag{46}$$

We now show that in the most regularized case, these no-intercept kernel models are actually performing kernel density classification. Initially, for $\lambda > \lambda_1$, we classify to class +1 if $f^*(x)/\lambda > 0$,

else to class -1. But

$$\begin{aligned} f^*(x) &= \sum_{j\in I_+} K(x,x_j) - \sum_{j\in I_-} K(x,x_j) \\ &= n\cdot\left(\frac{n_+}{n}\cdot\frac{1}{n_+}\sum_{j\in I_+} K(x,x_j) - \frac{n_-}{n}\cdot\frac{1}{n_-}\sum_{j\in I_-} K(x,x_j)\right) \quad (47) \\ &\propto \pi_+ h_+(x) - \pi_- h_-(x). \quad (48) \end{aligned}$$

In other words, this is the estimated Bayes decision rule, with h_+ the kernel density (Parzen window) estimate for the + class, π_+ the sample prior, and likewise for $h_-(x)$ and π_-. A similar observation is made in Schölkopf and Smola (2001), for the model with intercept. So at this end of the regularization scale, the kernel parameter γ plays a crucial role, as it does in kernel density classification. As γ increases, the behavior of the classifier approaches that of the 1-nearest neighbor classifier. For very small γ, or in fact a linear kernel, this amounts to closest centroid classification.

As λ is relaxed, the $\alpha_i(\lambda)$ will change, giving ultimately zero weight to points well within their own class, and sharing the weights among points near the decision boundary. In the context of nearest neighbor classification, this has the flavor of "editing", a way of thinning out the training set retaining only those prototypes essential for classification (Ripley, 1996).

All these interpretations get blurred when the intercept β_0 is present in the model.

For the radial kernel, a constant term is included in $\text{span}\{K(x,x_i)\}_1^n$, so it is not strictly necessary to include one in the model. However, it will get regularized (shrunk toward zero) along with all the other coefficients, which is usually why these intercept terms are separated out and freed from regularization. Adding a constant b^2 to $K(\cdot,\cdot)$ will reduce the amount of shrinking on the intercept (since the amount of shrinking of an eigenfunction of K is inversely proportional to its eigenvalue; see Section 5). For the linear SVM, we can augment the x_i vectors with a constant element b, and then fit the no-intercept model. The larger b, the closer the solution will be to that of the linear SVM with intercept.

7. Discussion

Our work on the SVM path algorithm was inspired by earlier work on exact path algorithms in other settings. "Least Angle Regression" (Efron et al., 2002) shows that the coefficient path for the sequence of "lasso" coefficients (Tibshirani, 1996) is piecewise linear. The lasso solves the following regularized linear regression problem,

$$\min_{\beta_0,\beta} \sum_{i=1}^n (y_i - \beta_0 - x_i^T\beta)^2 + \lambda|\beta|, \quad (49)$$

where $|\beta| = \sum_{j=1}^p |\beta_j|$ is the L_1 norm of the coefficient vector. This L_1 constraint delivers a sparse solution vector β_λ; the larger λ, the more elements of β_λ are zero, the remainder shrunk toward zero. In fact, any model with an L_1 constraint and a quadratic, piecewise quadratic, piecewise linear, or mixed quadratic and linear loss function, will have piecewise linear coefficient paths, which can be calculated exactly and efficiently for all values of λ (Rosset and Zhu, 2003). These models include, among others,

- A robust version of the lasso, using a "Huberized" loss function.

- The L_1 constrained support vector machine (Zhu et al., 2003).

The SVM model has a quadratic constraint and a piecewise linear ("hinge") loss function. This leads to a piecewise linear path in the dual space, hence the Lagrange coefficients α_i are piecewise linear.

Other models that would share this property include

- The ε-insensitive SVM regression model
- Quadratically regularized L_1 regression, including flexible models based on kernels or smoothing splines.

Of course, quadratic criterion + quadratic constraints also lead to exact path solutions, as in the classic case of ridge regression, since a closed form solution is obtained via the SVD. However, these paths are nonlinear in the regularization parameter.

For general non-quadratic loss functions and L_1 constraints, the solution paths are typically piecewise non-linear. Logistic regression is a leading example. In this case, approximate path-following algorithms are possible (Rosset, 2005).

The general techniques employed in this paper are known as parametric programming via active sets in the convex optimization literature (Allgower and Georg, 1992). The closest we have seen to our work in the literature employ similar techniques in incremental learning for SVMs (Fine and Scheinberg, 2002; Cauwenberghs and Poggio, 2001; DeCoste and Wagstaff, 2000). These authors do not, however, construct exact paths as we do, but rather focus on updating and downdating the solutions as more (or less) data arises. Diehl and Cauwenberghs (2003) allow for updating the parameters as well, but again do not construct entire solution paths. The work of Pontil and Verri (1998) recently came to our notice, who also observed that the lagrange multipliers for the margin vectors change in a piece-wise linear fashion, while the others remain constant.

The `SvmPath` has been implemented in the `R` computing environment (contributed library `svmpath` at CRAN), and is available from the first author's website.

Acknowledgments

The authors thank Jerome Friedman for helpful discussions, and Mee-Young Park for assisting with some of the computations. They also thank two referees and the associate editor for helpful comments. Trevor Hastie was partially supported by grant DMS-0204162 from the National Science Foundation, and grant RO1-EB0011988-08 from the National Institutes of Health. Tibshirani was partially supported by grant DMS-9971405 from the National Science Foundation and grant RO1-EB0011988-08 from the National Institutes of Health.

References

Eugene Allgower and Kurt Georg. Continuation and path following. *Acta Numerica*, pages 1–64, 1992.

Francis Bach and Michael Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS 2000)*, volume 13. MIT Press, Cambridge, MA, 2001.

Dennis DeCoste and Kiri Wagstaff. Alpha seeding for support vector machines. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 345–349. ACM Press, 2000.

Christopher Diehl and Gert Cauwenberghs. SVM incremental learning, adaptation and optimization. In *Proceedings of the 2003 International Joint Conference on Neural Networks*, pages 2685–2690, 2003. Special series on Incremental Learning.

Brad Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. Technical report, Stanford University, 2002.

Brad Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 2004. (to appear, with discussion).

Theodorus Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, Volume 13, Number 1, pages 1-50, 2000.

Shai Fine and Katya Scheinberg. Incas: An incremental active set method for SVM. Technical report, IBM Research Labs, Haifa, 2002.

Trevor Hastie and Robert Tibshirani. *Generalized Additive Models*. Chapman and Hall, 1990.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer Verlag, New York, 2001.

Trevor Hastie and Rob Tibshirani. Efficient quadratic regularization for expression arrays. Technical report, Stanford University, 2003.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

Thorsten Joachims. *Practical Advances in Kernel Methods — Support Vector Learning*, chapter Making large scale SVM learning practical. MIT Press, 1999. See `http://svmlight.joachims.org`.

Steve Marron. An overview of support vector machines and kernel methods. Talk, 2003. Available from author's website: `http://www.stat.unc.edu/postscript/papers/marron/Talks/`.

Massimiliano Pontil and Alessandro Verri. Properties of support vector machines. *Neural Computation*, 10(4):955–974, 1998.

S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. Mesirov, T. Poggio, W. Gerald, M. Loda, E. Lander, and T. Golub. Multiclass cancer diagnosis using tumor gene expression signature. *PNAS*, 98:15149–15154, 2001.

B. D. Ripley. Pattern recognition and neural networks. Cambridge University Press, 1996.

Saharon Rosset. Tracking curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems (NIPS 2004)*, volume 17. MIT Press, Cambridge, MA, 2005. to appear.

Saharon Rosset and Ji Zhu. Piecewise linear regularized solution paths. Technical report, Stanford University, 2003. http://www-stat.stanford.edu/∼saharon/papers/piecewise.ps.

Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin maximizing loss functions. In *Advances in Neural Information Processing Systems (NIPS 2003)*, volume 16. MIT Press, Cambridge, MA, 2004.

Bernard Schölkopf and Alex Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. MIT Press, 2001.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B.*, 58:267–288, 1996.

Vladimir Vapnik. *The Nature of Statistical Learning*. Springer-Verlag, 1996.

G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, 1990.

G. Wahba, Y. Lin, and H. Zhang. Gacv for support vector machines. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 297–311, Cambridge, MA, 2000. MIT Press.

J. Weston and C. Watkins. Multi-class support vector machines, 1998. URL citeseer.nj.nec.com/8884.html.

Christopher K. I. Williams and Matthias Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1159–1166. Morgan Kaufmann Publishers Inc., 2000.

Ji Zhu and Trevor Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 2004. (to appear).

Ji Zhu, Saharon Rosset, Trevor Hastie, and Robert Tibshirani. L1 norm support vector machines. Technical report, Stanford University, 2003.

Journal of Machine Learning Research 5 (2004) 1417–1433 Submitted 7/03; Published 10/04

Second Order Cone Programming Formulations for Feature Selection

Chiranjib Bhattacharyya CHIRU@CSA.IISC.ERNET.IN
Department of Computer Science and Automation
Indian Institute of Science
Bangalore, 560 012, India

Editor: John Shawe-Taylor

Abstract

This paper addresses the issue of feature selection for linear classifiers given the moments of the class conditional densities. The problem is posed as finding a minimal set of features such that the resulting classifier has a low misclassification error. Using a bound on the misclassification error involving the mean and covariance of class conditional densities and minimizing an L_1 norm as an approximate criterion for feature selection, a second order programming formulation is derived. To handle errors in estimation of mean and covariances, a tractable robust formulation is also discussed. In a slightly different setting the Fisher discriminant is derived. Feature selection for Fisher discriminant is also discussed. Experimental results on synthetic data sets and on real life microarray data show that the proposed formulations are competitive with the state of the art linear programming formulation.

1. Introduction

The choice of useful features for discriminating between two classes is an important problem and has many applications. This paper addresses the issue of constructing linear classifiers using a small number of features when data is summarized by its moments.

A linear two-class classifier is a function defined as

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^\top \mathbf{x} - b). \tag{1}$$

The classifier outputs 1 if the observation $\mathbf{x} \in \mathbb{R}^n$ falls in the halfspace $\{\mathbf{x}|\mathbf{w}^\top \mathbf{x} > b\}$, otherwise it outputs -1. During training, the parameters, $\{\mathbf{w}, b\}$, of the discriminating hyperplane $\mathbf{w}^\top \mathbf{x} = b$ are computed from a specified data set $D = \{(\mathbf{x}_i, y_i)|\mathbf{x}_i \in \mathbb{R}^n, y_i = \{1,-1\}, i = 1,\ldots,m\}$.

Finding useful features for linear classifiers is equivalent to searching for a $\mathbf{w}$, such that most elements of $\mathbf{w}$ are zero. This can be understood as when the ith component of $\mathbf{w}$ is zero, then by (1) the ith component of the observation vector $\mathbf{x}$ is irrelevant in deciding the class of $\mathbf{x}$. Using the L_0 norm of $\mathbf{w}$, defined as

$$\|\mathbf{w}\|_0 = |S| \;\; S = \{i|\mathbf{w}_i \neq 0\},$$

the problem of feature selection can be posed as a combinatorial optimization problem:

$$\begin{array}{ll} \underset{\mathbf{w},b}{\text{minimize}} & \|\mathbf{w}\|_0, \\ \text{subject to} & y_i\left(\mathbf{w}^\top \mathbf{x}_i - b\right) \geq 1, \;\; \forall 1 \leq i \leq m. \end{array} \tag{2}$$

The constraints ensure that the classifier correctly assigns labels to all training data points. Due to the unwieldy objective the formulation is intractable for large n (Amaldi and Kann, 1998). A heuris-

tic tractable approximation to the proposed objective is to minimize the L_1 norm of $\mathbf{w}$. For a discussion of this issue see Chen et al. (1999), and for other approximations to L_0 norm see Weston et al. (2003). In the sequel, we will enforce the feature selection criterion by minimizing the L_1 norm.

Let $\mathbf{X}_1$ and $\mathbf{X}_2$ denote n dimensional random vectors belonging to class 1 and class 2 respectively. Without loss of generality assume that class 1 is identified with the label $y = 1$, while class 2 is identified with label $y = -1$. Let the mean and covariance of $\mathbf{X}_1$ be $\mu_1 \in \mathbb{R}^n$ and $\Sigma_1 \in \mathbb{R}^{n\times n}$ respectively. Similarly for $\mathbf{X}_2$ the mean and covariance be $\mu_2 \in \mathbb{R}^n$ and $\Sigma_2 \in \mathbb{R}^{n\times n}$ respectively. Note that Σ_1 and Σ_2 are positive semidefinite symmetric matrices. In this paper we wish to address the problem of feature selection for linear classifiers given μ_1, μ_2, Σ_1 and Σ_2.

Lanckriet et al. (2002a,b) addressed the problem of classification given μ_1, μ_2, Σ_1 and Σ_2 in a minimax setting. In their approach, a Chebychev inequality is used to bound the error of misclassification. We wish to use the same inequality along with the L_1 norm minimization criterion for feature selection. This leads to a Second Order Cone Programming problem (SOCP). SOCPs are a special class of nonlinear convex optimization problems, which can be efficiently solved by interior point codes (Lobo et al., 1998). We also investigate a tractable robust formulation, which takes into account errors in estimating the moments.

The paper is organized as follows. In Section 2 the linear programming approach is discussed. The main contributions are in Section 3 and Section 4. The Chebychev bound and the feature selection criterion leads to a SOCP. The Fisher discriminant is also rederived using the Chebychev bound. We also discuss feature selection for the Fisher discriminant. A robust formulation is discussed in Section 4. Experimental results for these formulations are shown in Section 5. The concluding section summarizes the main contributions and future directions.

2. Linear Programming Formulation for Feature Selection

The problem of finding a $\{\mathbf{w}, b\}$, so that the hyperplane $\mathbf{w}^\top \mathbf{x} = b$ discriminates well between two classes and also selects a small number of features, can be posed by the following optimization problem.

$$\begin{array}{ll} \underset{\mathbf{w},b}{\text{minimize}} & \|\mathbf{w}\|_1, \\ \text{subject to} & y_i \left(\mathbf{w}^\top \mathbf{x}_i - b\right) \geq 1, \ \ \forall 1 \leq i \leq m. \end{array} \tag{3}$$

At optimality it is hoped that most of the elements of the weight vector $\mathbf{w}$ are zero. The above formulation can be posed as a Linear Programming (LP) problem by introducing two vectors in the following way.

$$\mathbf{w} = \mathbf{u} - \mathbf{v}; \ \ \|\mathbf{w}\|_1 = (\mathbf{u}+\mathbf{v})^\top \mathbf{e}; \ \ \mathbf{u} \geq 0, \ \ \mathbf{v} \geq 0. \tag{4}$$

This makes the nonlinear objective linear (see Fletcher, 1989) and the optimization problem can be posed as the following LP.

$$\begin{array}{ll} \underset{\mathbf{u},\mathbf{v},b}{\text{minimize}} & (\mathbf{u}+\mathbf{v})^\top \mathbf{e}, \\ \text{subject to} & y_i \left((\mathbf{u}-\mathbf{v})^\top \mathbf{x}_i - b\right) \geq 1 \ \ \forall 1 \leq i \leq m, \\ & \mathbf{u} \geq 0, \ \mathbf{v} \geq 0. \end{array} \tag{5}$$

The computational advantages of solving LPs make the above formulation extremely attractive. In the next section we discuss the problem of feature selection when data is summarized by the moments.

3. Feature Selection Using Moments

Let the data for each class be specified by the first two moments, the mean and covariance. The problem of feature selection, given the moments, is approached in a worst case setting by using a multivariate generalization of Chebychev-Cantelli inequality. The inequality is used to derive a SOCP, which yields a classifier using a very small number of features.

The following multivariate generalization of Chebychev-Cantelli inequality will be used in the sequel to derive a lower bound on the probability of a random vector taking values in a given half space.

Theorem 1 *Let* $\mathbf{X}$ *be a n dimensional random vector. The mean and covariance of* $\mathbf{X}$ *be* $\mu \in \mathbb{R}^n$ *and* $\Sigma \in \mathbb{R}^{n\times n}$. *Let* $\mathcal{H}(\mathbf{w},b) = \{\mathbf{z}|\mathbf{w}^\top \mathbf{z} < b, \mathbf{w},\mathbf{z} \in \mathbb{R}^n, b \in \mathbb{R}\}$ *be a given half space, with* $\mathbf{w} \neq 0$. *Then*

$$P(\mathbf{X} \in \mathcal{H}) \geq \frac{s^2}{s^2 + \mathbf{w}^\top \Sigma \mathbf{w}} \tag{6}$$

where $s = (b - \mathbf{w}^\top \mu)_+,\ (x)_+ = max(x,0)$.

For proof see Appendix A.

The theorem says that the probability of the event that an observation drawn from $\mathbf{X}$ takes values in the halfspace $\mathcal{H}$, can be bounded using μ and Σ. Let $\mathbf{X}_1 \sim (\mu_1, \Sigma_1)$ denote a class of distributions that have mean μ_1, and covariance Σ_1, but are otherwise arbitrary; likewise for class 2, $\mathbf{X}_2 \sim (\mu_2, \Sigma_2)$. The discriminating hyperplane tries to place class 1 in the half space $\mathcal{H}_1(\mathbf{w},b) = \{\mathbf{x}|\mathbf{w}^T\mathbf{x} > b\}$ and class 2 in the other half space $\mathcal{H}_2(\mathbf{w},b) = \{\mathbf{x}|\mathbf{w}^T\mathbf{x} < b\}$. To ensure this, one has to find $\{\mathbf{w},b\}$ such that $P(\mathbf{X}_1 \in \mathcal{H}_1)$ and $P(\mathbf{X}_2 \in \mathcal{H}_2)$ are both high. Lanckriet et al. (2002a,b) considers this problem and solves it in a minimax setting.

In this paper we consider the problem of feature selection. As remarked before, feature selection can be enforced by minimizing the L_1 norm of $\mathbf{w}$. To this end, consider the following problem

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_1, \\ s.t \quad & Prob(\mathbf{X}_1 \in \mathcal{H}_1) \geq \eta, \\ & Prob(\mathbf{X}_2 \in \mathcal{H}_2) \geq \eta, \\ & \mathbf{X}_1 \sim (\mu_1, \Sigma_1),\ \mathbf{X}_2 \sim (\mu_2, \Sigma_2). \end{aligned} \tag{7}$$

In most cases the objective yields a sparse $\mathbf{w}$. The two constraints state that the probability of belonging to the proper half space should be atleast more than a user defined parameter η. The parameter η takes values in $(0,1)$. Higher the value of η, more stringent is the requirement that all points belong to the correct half space.

The problem (7) has two constraints, one for each class, which states that the probability of a random vector taking values in a given half space is lower-bounded by η. These constraints can be posed as nonlinear constraints by applying theorem 1 (see Lanckriet et al., 2002b). The constraint for class 1 can be handled by setting

$$Prob(\mathbf{X}_1 \in \mathcal{H}_1) \geq \frac{(\mathbf{w}^T\mu_1 - b)_+^2}{(\mathbf{w}^T\mu_1 - b)_+^2 + \mathbf{w}^T\Sigma\mathbf{w}} \geq \eta,$$

which yield two constraints

$$\mathbf{w}^\top \mu_1 - b \geq \sqrt{\frac{\eta}{1-\eta}}\sqrt{\mathbf{w}^T \Sigma_1 \mathbf{w}}; \quad \mathbf{w}^\top \mu_1 - b \geq 0.$$

Similarly applying theorem 1 to the other constraint, two more constraints are obtained. Note that the constraints are positively homogenous, that is, if $\mathbf{w}, b$ satisfies the constraints then a $c\mathbf{w}, cb$ also satisfies the constraints where c is any positive number. To deal with this extra degree of freedom one can require that the classifier should separate μ_1 and μ_2 even if $\eta = 0$. One way to impose this requirement is via the constraint

$$\mathbf{w}^T \mu_1 - b \geq 1, \;\; b - \mathbf{w}^T \mu_2 \geq 1.$$

As both the matrices Σ_1 and Σ_2 are positive semi-definite, there exist matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ such that

$$\Sigma_1 = \mathbf{C}_1 \mathbf{C}_1^\top, \;\; \Sigma_2 = \mathbf{C}_2 \mathbf{C}_2^\top.$$

The problem (7) can now be stated as a deterministic optimization problem

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_1, \\ s.t \quad & \mathbf{w}^\top \mu_1 - b \geq \sqrt{\tfrac{\eta}{1-\eta}} \|\mathbf{C}_1^\top \mathbf{w}\|_2, \\ & b - \mathbf{w}^\top \mu_2 \geq \sqrt{\tfrac{\eta}{1-\eta}} \|\mathbf{C}_2^\top \mathbf{w}\|_2, \\ & \mathbf{w}^T \mu_1 - b \geq 1, \\ & b - \mathbf{w}^T \mu_2 \geq 1. \end{aligned}$$

The nonlinear objective can be tackled, as in (4), by introducing two vectors $\mathbf{u}$ and $\mathbf{v}$, which leads to the formulation

$$\begin{aligned} \min_{\mathbf{u},\mathbf{v},b} \quad & (\mathbf{u}+\mathbf{v})^\top \mathbf{e}, \\ s.t \quad & (\mathbf{u}-\mathbf{v})^\top \mu_1 - b \geq \sqrt{\tfrac{\eta}{1-\eta}} \|\mathbf{C}_1^\top (\mathbf{u}-\mathbf{v})\|_2, \\ & b - (\mathbf{u}-\mathbf{v})^\top \mu_2 \geq \sqrt{\tfrac{\eta}{1-\eta}} \|\mathbf{C}_2^\top (\mathbf{u}-\mathbf{v})\|_2, \\ & (\mathbf{u}-\mathbf{v})^\top \mu_1 - b \geq 1, \\ & b - (\mathbf{u}-\mathbf{v})^\top \mu_2 \geq 1, \\ & \mathbf{u} \geq 0, \mathbf{v} \geq 0. \end{aligned} \tag{8}$$

This problem is convex, and is an instance of SOCP. The nonlinear constraints are called Second Order Cone(SOC) constraints. A SOC constraint on the variable $\mathbf{x} \in \mathbb{R}^n$ is of the form

$$\mathbf{c}^\top \mathbf{x} + d \geq \|\mathbf{A}\mathbf{x} + \mathbf{b}\|_2,$$

where $\mathbf{b} \in \mathbb{R}^m, \mathbf{c} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{m \times n}$ are given. Minimizing a linear objective over SOC constraints is known as SOCP problems. Recent advances in interior point methods for convex nonlinear optimization (Nesterov and Nemirovskii, 1993) have made such problems feasible. As a special case of

convex nonlinear optimization SOCPs have gained much attention in recent times. For a discussion of efficient algorithms and applications of SOCP see Lobo et al. (1998).

On the training data set, the error rate of the classifier is upper bounded by $1-\eta$. This upper bound also holds for the generalization error (Lanckriet et al., 2002b) if the test data comes from a distribution having the same mean and covariance as the training data. As η is increased, the data is forced to lie on the correct side of the hyperplane with more probability. This should result in a smaller training error. Again with increasing η, sparseness would decrease, as more stress is given to accuracy. Thus the parameter η trades off accuracy with sparseness.

3.1 Feature Selection for Fisher Discriminants

In this section we derive the Fisher discriminant using the Chebychev bound and discuss a formulation for feature selection. For a linearly separable dataset, one can find $\{\mathbf{w}, b\}$ so that all observations belonging to class 1(class 2) obey $\mathbf{w}^\top \mathbf{x}_1 \geq b$ ($\mathbf{w}^\top \mathbf{x}_2 \leq b$), which implies $\mathbf{w}^\top \mathbf{X}_1 \geq b(\mathbf{w}^\top \mathbf{X}_2 \leq b)$. If $\mathbf{X} = \mathbf{X}_1 - \mathbf{X}_2$ defines the difference between the class conditional random vectors, then $\mathbf{X}$ lies in the halfspace $\mathcal{H}(\mathbf{w}) = \{\mathbf{z} | \mathbf{w}^\top \mathbf{z} \geq 0\}$. One can derive the Fisher discriminant by considering the following formulation

$$\begin{aligned} \max_{\mathbf{w},\eta} \quad & \eta \\ s.t \quad & Prob(\mathbf{X} \in \mathcal{H}) \geq \eta, \quad \mathbf{X} \sim (\mu, \Sigma). \end{aligned} \tag{9}$$

As $\mathbf{X}_1$ and $\mathbf{X}_2$ are independent the mean of $\mathbf{X}$ is $\mu = \mu_1 - \mu_2$ and covariance $\Sigma = \Sigma_1 + \Sigma_2$. Using the Chebychev bound (6) the constraint can be lower bounded by

$$Prob(\mathbf{X} \in \mathcal{H}) \geq \frac{(\mathbf{w}^\top \mu)^2}{(\mathbf{w}^\top \mu)^2 + \mathbf{w}^\top \Sigma \mathbf{w}}; \quad \mathbf{w}^\top \mu \geq 0,$$

and hence it follows that (9) is equivalent to solving

$$\max_{\mathbf{w}} \frac{\{\mathbf{w}^\top (\mu_1 - \mu_2)\}^2}{\mathbf{w}^\top (\Sigma_1 + \Sigma_2)\mathbf{w}}, \tag{10}$$

which is same as the Fisher discriminant. The above formulation shows that Fisher discriminant can be understood as computing a discriminant hyperplane whose generalization error is less than $1-\eta^*$, where

$$\eta^* = \frac{d(\mathbf{w}^*)}{1 + d(\mathbf{w}^*)}; \quad d(\mathbf{w}^*) = \max_{\mathbf{w}} \frac{\{\mathbf{w}^\top (\mu_1 - \mu_2)\}^2}{\mathbf{w}^\top (\Sigma_1 + \Sigma_2)\mathbf{w}}.$$

The bound holds provided the data distribution has the neccessary first and second moments. One can incorporate feature selection by minimizing the L_1 norm of $\mathbf{w}$ for a fixed value of η as follows

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_1 \\ s.t \quad & Prob(\mathbf{X} \in \mathcal{H}) \geq \eta, \quad \mathbf{X} \sim (\mu_1 - \mu_2, \Sigma_1 + \Sigma_2) \end{aligned} \tag{11}$$

and arguing as in (8) the following SOCP

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_1, \\ s.t \quad & \mathbf{w}^\top(\mu_1-\mu_2) \geq \sqrt{\tfrac{1-\eta}{\eta}}\sqrt{\mathbf{w}^\top(\Sigma_1+\Sigma_2)\mathbf{w}}, \\ & \mathbf{w}^\top(\mu_1-\mu_2) \geq 1, \end{aligned} \tag{12}$$

is obtained. The parameter η ensures that the resulting classifier has a misclassification error less than $1-\eta$, while feature selection is ensured by the objective.

3.2 Estimation of Mean and Covariance for Each Class

Let $T_1 = [\mathbf{x}_{11},\ldots,\mathbf{x}_{1m_1}]$ be the data matrix for one class, say with label $y = 1$. Similarly $T_2 = [\mathbf{x}_{21},\ldots,\mathbf{x}_{2m_2}]$ be the data matrix for the other class having the label $y = -1$. Both the matrices have the same number of rows, n, the number of features. The columns correspond to data points; m_1 data points for the first class and m_2 data points for the other class. For microarray data sets the number of features, n, is in thousands, while the number of examples, m_1 or m_2, is less than hundred.

In the present formulation, empirical estimates of the mean and covariance are used

$$\mu_1 = \overline{\mathbf{x}}_1 = \frac{1}{m_1}T_1\mathbf{e},\ \ \mu_2 = \overline{\mathbf{x}}_2 = \frac{1}{m_2}T_2\mathbf{e};$$

$$\Sigma_1 = \overline{\Sigma}_1 = \mathbf{C}_1\mathbf{C}_1^T,\ \ \mathbf{C}_1 = \frac{1}{\sqrt{m_1}}(T_1 - \mu_1\mathbf{e}^\top);$$

$$\Sigma_2 = \overline{\Sigma}_2 = \mathbf{C}_2\mathbf{C}_2^T,\ \ \mathbf{C}_2 = \frac{1}{\sqrt{m_2}}(T_2 - \mu_2\mathbf{e}^\top).$$

Note that the covariances are huge matrices (of size $n \times n$). Instead of storing such huge matrices one can store the much smaller matrices C_1 and C_2 of size $n \times m_1$ and $n \times m_2$ respectively. The resulting classifier is heavily dependent on the estimates of the mean and covariance. In the next section, we will discuss classifiers which are robust to errors in the estimation of mean and covariance.

4. A Robust Formulation

In practical cases it might happen that the error rate of the classifiers is well above $1-\eta$. As pointed out by Lanckriet et al. (2002b), this problem often occurs when the training data set has very few data-points compared to the number of features, for example, microarray data sets. In such cases the estimates of mean and covariance are not very accurate. It will be useful, especially for microarray data sets, to explore formulations which can yield classifiers robust to such estimation errors. In the following, we discuss one such formulation.

We assume that the means and covariances take values in a specified set, in particular $(\mu_1,\Sigma_1) \in U_1$ where $U_1 \subset \mathbb{R}^n \times S_n^+$ and S_n^+ is the set of all positive semidefinite $n \times n$ matrices. Similarly another set U_2 is defined which characterizes the values of (μ_2,Σ_2). Consider the robust version of

formulation (7),

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|_1, \\ s.t \quad & Prob(\mathbf{X}_1 \in \mathcal{H}_1) \geq \eta, \\ & Prob(\mathbf{X}_2 \in \mathcal{H}_2) \geq \eta, \\ & \mathbf{X}_1 \sim (\mu_1, \Sigma_1)\ \mathbf{X}_2 \sim (\mu_2, \Sigma_2), \\ & (\mu_1, \Sigma_1) \in U_1,\ (\mu_2, \Sigma_2) \in U_2. \end{aligned} \tag{13}$$

It ensures that the misclassification rate of the classifier is always less than $1-\eta$, for any arbitrary distribution whose means and covariances take values in some specified sets.

The tractability of this formulation depends on the definition of the sets U_1 and U_2. We assume that the sets describing the values of means and covariances are independent of one another, more precisely U_{m1}, U_{m2}, U_{v1} and U_{v2} describe the uncertainty in the values of μ_1, μ_2, Σ_1 and Σ_2 respectively. As before, applying the Chebychev bound and with a reformulation of (8) the following robust version

$$\begin{aligned} \min_{\mathbf{w},b,t_1,t_2} \quad & \|\mathbf{w}\|_1 \\ s.t. \quad & \mathbf{w}^\top \mu_1 - b \geq t_1\ \forall \mu_1 \in U_{m1}, \\ & b - \mathbf{w}^\top \mu_2 \geq t_2\ \forall \mu_2 \in U_{m2}, \\ & \sqrt{\mathbf{w}^\top \Sigma_1 \mathbf{w}} \leq \sqrt{\tfrac{1-\eta}{\eta}} t_1\ \forall \Sigma_1 \in U_{v1}, \\ & \sqrt{\mathbf{w}^\top \Sigma_2 \mathbf{w}} \leq \sqrt{\tfrac{1-\eta}{\eta}} t_2\ \forall \Sigma_2 \in U_{v2}, \\ & t_1 \geq 1,\ t_2 \geq 1 \end{aligned} \tag{14}$$

is obtained. The reformulation is obtained by modifying the SOC constraint corresponding to class 1 by introducing a new variable t_1 as follows,

$$\mathbf{w}^\top \mu_1 - b \geq t_1 \geq \sqrt{\frac{\eta}{1-\eta}} \|\mathbf{C}_1^\top \mathbf{w}\|_2,\ t_1 \geq 1.$$

Likewise another variable is introduced to deal with the other SOC constraint belonging to class 2. To restrict the uncertainty to a low dimension space the following assumption is made.

Assumption 1 The random vector $\mathbf{X}_1$ takes values in the linear span of columns of T_1, while the random vector $\mathbf{X}_2$ takes values in the linear span of columns of T_2. More precisely the n dimensional random vectors $\mathbf{X}_1$ and $\mathbf{X}_2$ are linearly related to a m_1 dimensional random vector $\mathbf{Z}_1$ and a m_2 dimensional random vector$\mathbf{Z}_2$ respectively as follows

$$\mathbf{X}_1 = T_1 \mathbf{Z}_1, \quad \mathbf{X}_2 = T_2 \mathbf{Z}_2. \tag{15}$$

For microarray data sets m_1 and m_2 are much smaller than n. Thus, the assumption restricts the random variables $\mathbf{X}_1$ and $\mathbf{X}_2$ to much smaller dimension spaces. Let μ_{z1}, Σ_{z1} be the mean and covariance of the random variable $\mathbf{Z}_1$, and μ_{z2}, Σ_{z2} be the mean and covariance of the random variable $\mathbf{Z}_2$. It follows that

$$\mu_1 = T_1 \mu_{z1},\ \mu_2 = T_2 \mu_{z2};\ \Sigma_1 = T_1 \Sigma_{z1} T_1^\top,\ \Sigma_2 = T_2 \Sigma_{z2} T_2^\top.$$

Clearly then the sample estimates $\overline{\mathbf{z}}_1$ and $\overline{\mathbf{z}}_2$ are related to $\overline{\mathbf{x}}_1$ and $\overline{\mathbf{x}}_2$ by

$$\overline{\mathbf{x}}_i = T_i\overline{\mathbf{z}}_i \quad \overline{\Sigma}_i = T_i\overline{\Sigma}_{zi}T_i^\top, \quad \overline{\Sigma}_{zi} = \frac{1}{m_i}(\mathbf{I}-\mathbf{e}\mathbf{e}^\top)\forall i \in \{1,2\}. \tag{16}$$

Assuming an ellipsoidal uncertainty on the estimate of μ_1 and in light of (16), we define

$$U_{m1} = \{\mu_1 | \mu_1 = T_1\mu_{z1}, \quad (\mu_{z1}-\overline{\mathbf{z}}_1)^\top T_1^\top T_1(\mu_{z1}-\overline{\mathbf{z}}_1) \leq \delta^2\}.$$

For the uncertainty set U_{m1} and a given $\mathbf{w}$ the constraint

$$\mathbf{w}^\top\mu_1 - b \geq t_1 \ \forall \mu_1 \in U_{m1},$$

is equivalent to

$$\min_{\mu_1 \in U_{m1}} \mathbf{w}^\top\mu_1 - b \geq t_1. \tag{17}$$

Noting that minimizing a linear function over an ellipsoid is a convex optimization problem having the following closed form solution (see Appendix B),

$$\min_{\mu_1 \in U_{m1}} \mathbf{w}^\top\mu_1 = \frac{1}{m_1}\mathbf{w}^\top T_1\mathbf{e} - \delta\|T_1\mathbf{w}\|, \tag{18}$$

the constraint (17) can be restated as

$$\frac{1}{m_1}\mathbf{w}^\top T_1\mathbf{e} - b \geq t_1 + \delta\|T_1\mathbf{w}\|. \tag{19}$$

Similarly for μ_2 the uncertainty set is defined as

$$U_{m2} = \{\mu_2 | \mu_2 = T_2\mu_{z2}, \quad (\mu_{z2}-\overline{\mathbf{z}}_2)^\top T_2^\top T_2(\mu_{z2}-\overline{\mathbf{z}}_2) \leq \delta^2\},$$

and analogous to (19) the following constraint

$$b - \frac{1}{m_2}\mathbf{w}^\top T_2\mathbf{e} \geq t_2 + \delta\|T_2\mathbf{w}\|, \tag{20}$$

is obtained. Following Lanckriet et al. (2002b), the sets characterizing the covariance matrices are defined using Frobenius norm

$$U_{vi} = \{\Sigma_i | \Sigma_i = T_i\overline{\Sigma}_{zi}T_i^\top \ \ \|\Sigma_{zi} - \overline{\Sigma}_{zi}\|_F \leq \rho\} \ \ i = \{1,2\}.$$

Imposing robustness to estimation errors in the covariance matrix Σ_i is equivalent to the constraint

$$\max_{\Sigma_i \in U_{vi}} \sqrt{\mathbf{w}^\top\Sigma_i\mathbf{w}} \leq \sqrt{\frac{1-\eta}{\eta}}t_i, \ \ i = \{1,2\}. \tag{21}$$

Using the result (see Appendix B)

$$\max_{\Sigma_i \in U_{vi}} \sqrt{\mathbf{w}^\top\Sigma_i\mathbf{w}} = \sqrt{\mathbf{w}^\top T_i(\overline{\Sigma}_{zi} + \rho\mathbf{I})T_i^\top\mathbf{w}},$$

η	0	0.2	0.4	0.6	0.8	0.9	0.95	0.99
fs	10	10	10	10	10	10	9, 10	7,8,9,10

Table 1: The set of selected features, fs, for various values of η on synthetic data set. See text for more details.

the formulation (14) turns out to be

$$\begin{aligned} \min_{\mathbf{w},b,t_1,t_2} \quad & \|\mathbf{w}\|_1, \\ s.t. \quad & \tfrac{1}{m_1}\mathbf{w}^\top T_1\mathbf{e} - b \geq t_1 + \delta\|T_1\mathbf{w}\|_2, \\ & b - \tfrac{1}{m_2}\mathbf{w}^\top T_2\mathbf{e} \geq t_2 + \delta\|T_2\mathbf{w}\|_2, \\ & \|\mathbf{C}_{1z}^\top T_1^\top \mathbf{w}\|_2 \leq \sqrt{\tfrac{1-\eta}{\eta}}t_1, \\ & \|\mathbf{C}_{2z}^\top T_2^\top \mathbf{w}\|_2 \leq \sqrt{\tfrac{1-\eta}{\eta}}t_2, \\ & t_1 \geq 1 \;\; t_2 \geq 1, \end{aligned} \tag{22}$$

an SOCP. The matrix $\mathbf{C}_{1z}$ is obtained by using the cholesky decomposition of the regularized matrix $\overline{\Sigma}_{z1} + \rho I$, similarly for $\mathbf{C}_{2z}$.

As a consequence of Assumption 1, one needs to factorize matrices of size $m_1 \times m_1$, and $m_2 \times m_2$ instead of a much larger $n \times n$ matrix for the Frobenius norm uncertainty model. Thus, the assumption has computational benefits but it is quite restrictive. However, in absence of any prior knowledge, this maybe a good alternative to explore. In the next section, we experiment on the formulations (8) and (22) on both synthetic and real world microarray data sets.

5. Experiments

The feature selection abilities of the proposed formulations were tested on both synthetic and real world data sets. As a benchmark, the performance of the LP formulation on the same data sets are also reported.

Consider a synthetic data set generated as follows. The class label,y, of each observation was randomly chosen to be 1 or -1 with probability 0.5. The first ten features of the observation, $\mathbf{x}$, are drawn as $y\mathcal{N}(-i,1)$, where $\mathcal{N}(\mu,\sigma^2)$ is a gaussian centered around μ and variance σ^2. Nine hundred ninety other features were drawn as $\mathcal{N}(0,1)$. Fifty such observations were generated. The feature selection problem is to detect the first ten features, since they are the most discriminatory from the given pool of 1000 features, when the sample size is fifty.

The features were selected by the following procedure (see **?**). From the data set fifty partitions was generated by holding out one example as test data and others as training data. For each partiton, formulation (8) was solved on the training set for a fixed value of η using the open source package SEDUMI (Sturm, 1999) to obtain a set of features and the resulting classifier was used to predict the label of test data. The union of fifty sets of features is reported in Table 1 for various values of η. The average number of errors on all the fifty test sets, the Leave one out(LOO) error, was found to be 0. For low values of η say $\eta = 0.2$, only one feature, feature number 10, was selected. This is not surprising because among the ten features, feature number 10 has the most discriminatory

power. As the value of η is increased the formulation reports more discriminatory features. For $\eta = 0.95$, the formulation reported $\{7,8,9,10\}$, a set of four features. This shows that inspite of sample size being low compared to the number of features this formulation is able to discover the most discriminatory features. The corresponding list of features selected by the LP(5) is fs $=$ $\{2,3,4,5,6,7,8,9,10\}$. Both the formulations pick up the most discriminatory features but the LP formulation picks up more features. The experiment was repeated for 100 randomly generated data sets, and gave similar results. This demonstrates that the formulation (8) picks up discriminatory features and is comparable to the LP formulation.

We also experimented with the robust formulation (14) for different values of δ, and ρ. In Figure 1 number of features are plotted for various values of δ. As δ increases, the number of features selected by the formulation (22) increases, the value of ρ was zero for the reported experiment. The robust formulation tries to maintain the classification accuracy even when the estimates of mean and covariance are not correct. To ensure this, more features are needed to maintain the accuracy. Similar results were also obtained by varying ρ.

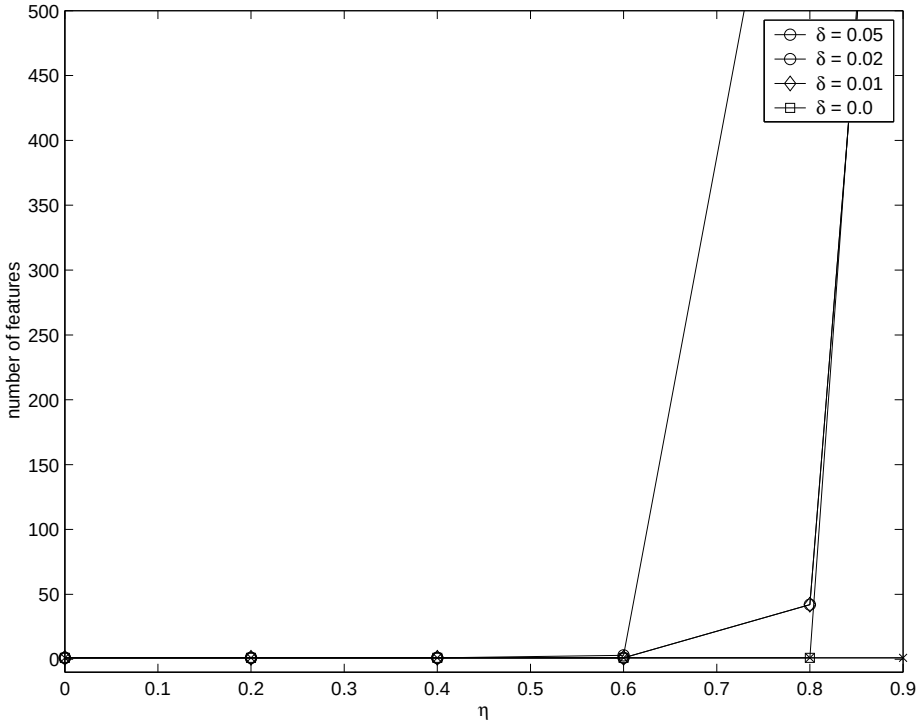


Figure 1: Plots of number of selected features versus η for various values of δ.

The formulation (8) was tested on six data sets (B, C, D, E, F, G) defined by Bhattacharyya et al. (2003). These binary classification problems are related to Small Round Blue Cell Tumors (SRBCT). Each data set have various number of data points but have the same number of features, $n = 2308$.

From the given data set, a partition was generated by holding out a data point as test set while the training set consisted of all the other data points. For each fixed value of η, the formulation

data set	B	C	D	E	F	G
SOCP	38	46	25	1	23	21
LP	21	8	8	2	12	13

Table 2: Number of selected features for which the LOO error was minimum. The row titled SOCP tabulates the minimum number of features reported by (8) for which the LOO error was zero. The row titled LP tabulates the number of features selected by the LP formulation. Total number of features is $n = 2308$

was solved for all possible partitions. For each partition the resulting classifier was tested on the held out data point. Average number of errors over all the partitions was reported as the LOO (leave one out) error. The results were compared against the linear programming formulation(5). Table 2 compares the number of features required to attain a zero LOO error by formulations (8) and (5). In both cases very small number of features, less than 2% of the total number of 2308 features, were selected. However the LP formulations almost always found a smaller set of features. Figures 2, 3, 4 show plots for the number of features selected by (8) for various values of η. As η increases, the number of features increase. For comparison, the number of features selected by the LP formulation is also plotted on the same graph. Figures 5, 6, 7 show plots for the LOO error the SOCP formulation. As η increases the LOO error decreases. This conforms to the view that as η increases, the classifier is forced to be accurate which leads to increase in the number of features.

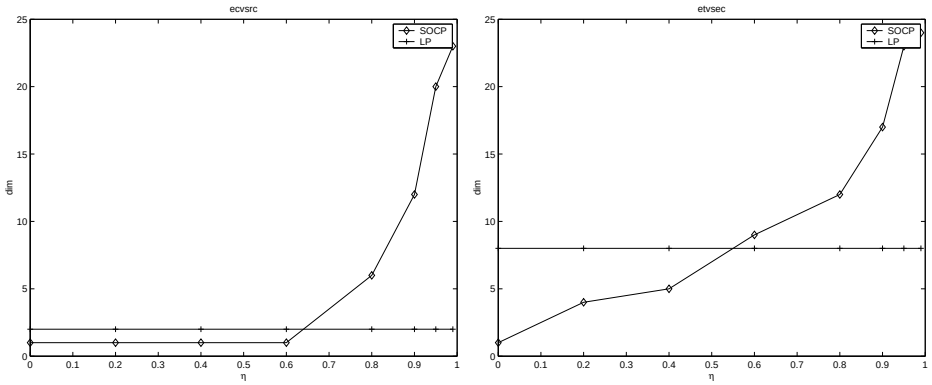


Figure 2: Plots of number of selected features versus η for data sets E, F.

6. Conclusions and Future Directions

The problem of selecting discriminatory features by using the moments of the class conditional densities was addressed in the paper. Using a Chebyshev-Cantelli inequality, the problem was posed as a SOCP. The above approach was also used to derive a formulation for doing feature selection for Fisher discriminants. A robust formulation was discussed which yields classifiers robust to estimation errors in the mean and covariance.

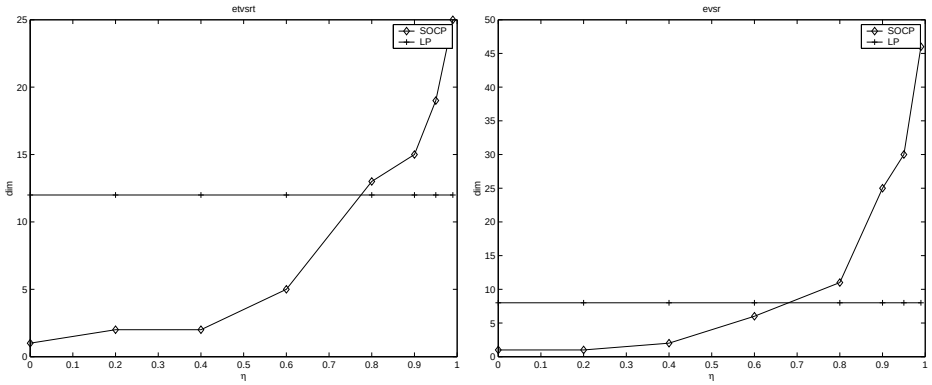


Figure 3: Plots of number of selected features versus η for data sets D, C.

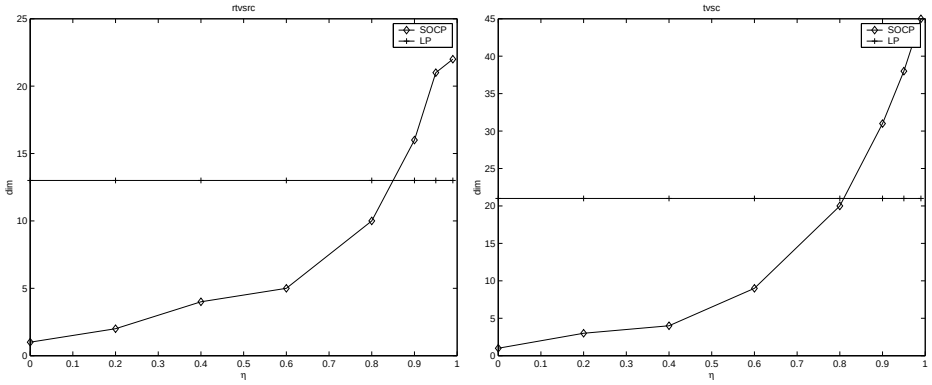


Figure 4: Plots of number of selected features versus η for data sets G, B.

On a toy data set, the formulation discovered the discriminatory features. The formulation has a parameter η, which can trade off accuracy with number of features. For small values of η, low number of discriminatory features are reported, while as η is increased the formulation reports more number of features. As borne out by experiments on the microarray data sets, the accuracy of the classifier increases as η is increased.

The approach in this paper can also be extended to design nonlinear classifiers using very few support vectors. Let the discriminating surface be $\{\mathbf{x}|\sum_i \alpha_i K(\mathbf{x}_i,\mathbf{x}) = b\}$ which divides the n-dimensional Euclidean space into two disjoint subsets $\{\mathbf{x}|\sum_i \alpha_i K(\mathbf{x},\mathbf{x}_i) < b\}$ and $\{\mathbf{x}|\sum_i \alpha_i K(\mathbf{x},\mathbf{x}_i) > b\}$, where the kernel K, is a function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ obeying the Mercer conditions (Mercer, 1909).

One can restate the nonlinear discriminating surface by a hyperplane in m dimensions,

$$\mathcal{H} = \{\mathbf{x}|\alpha^\top k(\mathbf{x}) - b = 0\}$$

where m is the number of examples and $k(\mathbf{x})$ is a vector in m dimensions whose ith component is $k(\mathbf{x},\mathbf{x}_i)$. The set of support vectors is defined by $S = \{i|\alpha_i \neq 0\}$. We wish to find a decision

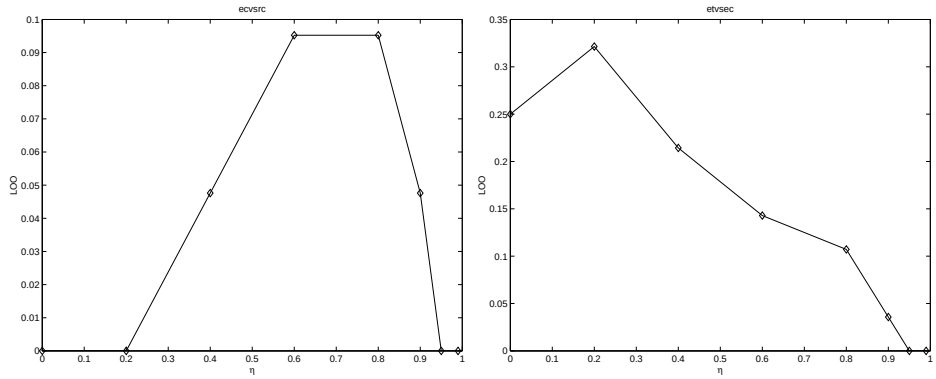


Figure 5: Plots of LOO error versus η for data sets E, F.

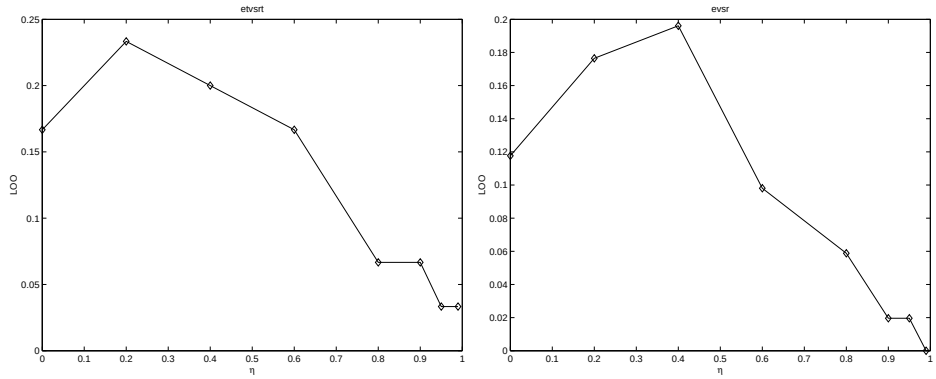


Figure 6: Plots of LOO error versus η for data sets D, C.

surface utilizing very small number of these vectors, or in other words, the goal is to minimize the cardinality of the set S, which can be approximated by the L_1 norm of α.

Let $k_1 = k(\mathbf{X}_1)$ be a random vector corresponding to class 1 while $k_2 = k(\mathbf{X}_2)$ be another random vector belonging to class 2. Let the means of k_1 and k_2 be $\tilde{k}_1$ and $\tilde{k}_2$ respectively and the covariance be $\tilde{\Sigma}_1$ and $\tilde{\Sigma}_2$ respectively. The problem can be approached as in (7) and on applying the Chebychev bound (6) the following formulation

$$\begin{aligned} \min_{\alpha,b} \quad & \|\alpha\|_1, \\ s.t \quad & \alpha^\top \tilde{k}_1 - b \geq \sqrt{\tfrac{\eta}{1-\eta}}\sqrt{\alpha^\top \tilde{\Sigma}_1^\top \alpha}, \\ & b - \alpha^\top \tilde{k}_2 \geq \sqrt{\tfrac{\eta}{1-\eta}}\sqrt{\alpha^\top \tilde{\Sigma}_2^\top \alpha}, \\ & \alpha^\top \tilde{k}_1 - b \geq 1, \\ & b - \alpha^\top \tilde{k}_2 \geq 1, \end{aligned} \tag{23}$$

is obtained. We believe this can have non-trivial advantages for data-mining problems.

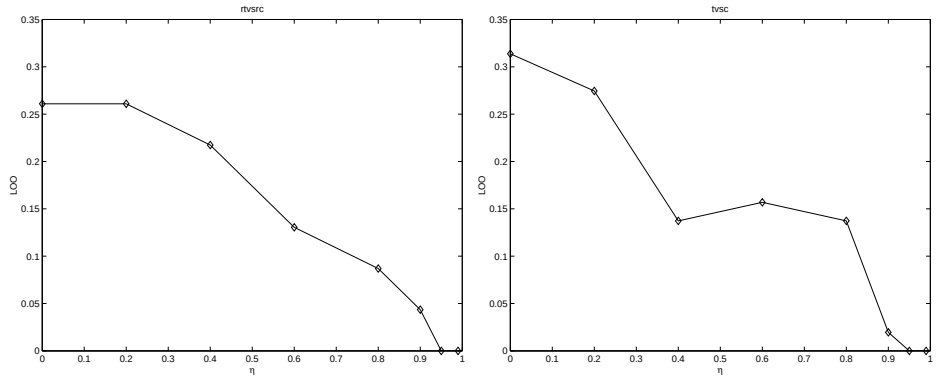


Figure 7: Plots of LOO error versus η for data sets G, B.

Acknowledgments

The author thanks the referees and the editor for their constructive comments. Their suggestions improved the paper significantly.

Appendix A. The Chebychev-Cantelli inequality

In this appendix we prove a multivariate generalization of the one sided Chebychev inequality. This inequality will be used to derive a lower bound on the probability of a multivariate random variable taking values in a given half space. Marshall and Olkin (1960)(also see Popescu and Bertsimas, 2001) proved a more general case.

We first state and prove the Chebychev inequality for a univariate random variable, (Lugosi, 2003). It would be useful to recall the Cauchy-Schwartz inequality. Let X and Y be random variables with finite variances, $\mathbb{E}(X-\mathbb{E}(X))^2 < \infty$ and $\mathbb{E}(Y-\mathbb{E}(Y))^2 < \infty$, then

$$|\mathbb{E}[(X-\mathbb{E}(X))(Y-\mathbb{E}(Y))]| \leq \sqrt{\mathbb{E}(X-\mathbb{E}(X))^2\mathbb{E}(Y-\mathbb{E}(Y))^2}.$$

Chebychev-Cantelli inequality *Let $s \geq 0$, Then*

$$P(X-\mathbb{E}(X) < s) \geq \frac{s^2}{s^2+\mathbb{E}(X-\mathbb{E}(X))^2}.$$

Proof Let $Y = X-\mathbb{E}(X)$. Note that $\mathbb{E}(Y) = 0$. For any s,

$$s = \mathbb{E}(s-Y) \leq \mathbb{E}((s-Y)I_{\{Y<s\}}(Y)).$$

For any $s \geq 0$, using the Cauchy-Schwartz inequality

$$\begin{aligned} s^2 \leq & \quad \mathbb{E}(s-Y)^2\mathbb{E}(I^2_{\{Y<s\}}(Y)), \\ = & \quad \mathbb{E}(s-Y)^2P(Y<s), \\ = & \quad (\mathbb{E}(Y^2)+s^2)P(Y<s). \end{aligned} \tag{24}$$

On rearranging terms one obtains

$$P(Y < s) \geq \frac{s^2}{\mathbb{E}(Y^2) + s^2}$$

and the result follows. ■

The above inequality can be used to derive a lower bound on the probability of a random vector taking values in a given half space.

Theorem 1 *Let* $\mathbf{X}$ *be a n dimensional random vector. The mean and covariance of* $\mathbf{X}$ *be* $\mu \in \mathbb{R}^n$ *and* $\Sigma \in \mathbb{R}^{n \times n}$. *Let* $\mathcal{H}(\mathbf{w}, b) = \{\mathbf{z} | \mathbf{w}^\top \mathbf{z} < b, \mathbf{w}, \mathbf{z} \in \mathbb{R}^n, b \in \mathbb{R}\}$ *be a given half space, with* $\mathbf{w} \neq 0$. *Then*

$$P(\mathbf{X} \in \mathcal{H}) \geq \frac{s^2}{s^2 + \mathbf{w}^\top \Sigma \mathbf{w}},$$

where $s = (b - \mathbf{w}^\top \mu)_+$, $(x)_+ = max(x, 0)$.

Proof There are two cases: $b \leq \mathbf{w}^\top \mu$ and $b > \mathbf{w}^\top \mu$.

For the case $b \leq \mathbf{w}^T \mu$, $s = 0$, and plugging its value in the Chebychev-Cantelli inequality, yields $P(\mathbf{X} \in \mathcal{H}) \geq 0$, which is trivially true. For the other case $b > \mathbf{w}^T \mu$, by definition $s = b - \mathbf{w}^T \mu$. Define $Y = \mathbf{w}^\top \mathbf{x}$, so that $\mathbb{E}(Y) = \mathbf{w}^T \mu$ $\mathbb{E}(Y - \mathbb{E}(Y))^2 = \mathbf{w}^\top \Sigma \mathbf{w}$. We have

$$P(\mathbf{X} \in \mathcal{H}) = P(Y < b) = P(Y - \mathbb{E}(Y) < s).$$

Application of Chebyshev-Cantelli inequality to the above relationship gives our desired result. This completes the proof. ■

Note that the proof does not require Σ to be invertible. For a more general proof pertaining to convex sets and tightness of the bound see (Marshall and Olkin, 1960, Popescu and Bertsimas, 2001).

Appendix B. Uncertainty in Covariance Matrices

Consider the following problem

$$\begin{aligned} &\max_\Sigma \sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}, \\ &\Sigma = T \Sigma_z T^\top, \\ &\|\Sigma_z - \overline{\Sigma}_z\|_F \leq \rho. \end{aligned} \tag{25}$$

Eliminating the equality constraint the objective can be stated as

$$\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}} = \sqrt{\mathbf{w}^\top T \Sigma_z T^\top \mathbf{w}}.$$

Introduce a new variable $\Delta\Sigma \in S_n^+$, so that $\Sigma_z = \overline{\Sigma}_z + \Delta\Sigma$, and the optimization problem (25) can be stated as

$$\begin{aligned} \max_{\Delta\Sigma} \quad & \sqrt{\mathbf{w}^\top T (\overline{\Sigma}_z + \Delta\Sigma) T^\top \mathbf{w}}, \\ s.t. \quad & \|\Delta\Sigma\|_F \leq \rho. \end{aligned} \tag{26}$$

The optimal is achieved at $\Delta\Sigma = \rho I$, see Appendix C in Lanckriet et al. (2002b) for a proof.

B.1 Notation

The vector $[1,1,\ldots,1]^\top$ will denoted by $\mathbf{e}$, and $[0,\ldots,0]$ by 0, the dimension will be clear from the context. If $\mathbf{w} = [w_1,\ldots,w_n]^\top$, we write $\mathbf{w} \geq 0$ to mean $w_i \geq 0, \forall i \in \{1,\ldots,n\}$. The euclidean norm of a vector $\mathbf{x} = [x_1,\ldots,x_n]^\top$, will be denoted by $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$, while the 1-norm of $\mathbf{x}$ will be denoted by $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$. The indicator function defined on the set A, denoted by $I_A(x)$, is

$$I_A(x) = \begin{cases} 1 & x \in A \\ 0 & otherwise. \end{cases}$$

The cardinality of set A is given by $|A|$. The Frobenius norm of a $m \times n$ matrix A, is denoted by $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2}$.

References

E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 290:237–260, 1998.

C. Bhattacharyya, L. Grate, A. Rizki, D. Radisky, F. Molina, M. I. Jordan, M. Bissell, and Saira I. Mian. Simultaneous classification and relevant feature identification in high-dimensional spaces: application to molecular profiling data. *Signal Processing*, 83:729–743, 2003.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *Siam Journal of Scientific Computing*, 20(1):33–61, 1999.

R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, New York, 1989.

G. R. G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan. Minimax probability machine. In *Advances in Neural Information Processing Systems*. MIT Press, 2002a.

G. R. G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, pages 555 – 582, December 2002b.

M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1–3):193–228, 1998.

G. Lugosi. Concentration-of-measure inequalities, 2003. URL `http://www.econ.upf.es/~lugosi/pre.html/anu.ps`. Lecture notes presented at the Machine learning Summer School 2003, ANU, Canberra.

A. W. Marshall and I. Olkin. Multivariate Chebychev inequalities. *Annals of Mathematical Statistics*, 31(4):1001–1014, 1960.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415–446, 1909.

Y. Nesterov and A. Nemirovskii. *Interior Point Algorithms in Convex Programming*. Number 13 in Studies in Applied Mathematics. SIAM, Philadelphia, 1993.

I. Popescu and D. Bertsimas. Optimal inequalities in probability theory. Technical Report TM 62, INSEAD, 2001.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11/12(1-4):625–653, 1999.

J. Weston, A. Elisseff, B. Schölkopf, and M. Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3, 2003.

Journal of Machine Learning Research 5 (2004) 1435–1455 Submitted 1/04; Revised 7/04; Published 11/04

Fast String Kernels using Inexact Matching for Protein Sequences

Christina Leslie CLESLIE@CS.COLUMBIA.EDU
Center for Computational Learning Systems
Columbia University
New York, NY 10115, USA

Rui Kuang RKUANG@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
New York, NY 10027, USA

Editor: Kristin Bennett

Abstract

We describe several families of k-mer based string kernels related to the recently presented mismatch kernel and designed for use with support vector machines (SVMs) for classification of protein sequence data. These new kernels – restricted gappy kernels, substitution kernels, and wildcard kernels – are based on feature spaces indexed by k-length subsequences ("k-mers") from the string alphabet Σ. However, for all kernels we define here, the kernel value $K(x,y)$ can be computed in $O(c_K(|x|+|y|))$ time, where the constant c_K depends on the parameters of the kernel but is independent of the size $|\Sigma|$ of the alphabet. Thus the computation of these kernels is linear in the length of the sequences, like the mismatch kernel, but we improve upon the parameter-dependent constant $c_K = k^{m+1}|\Sigma|^m$ of the (k,m)-mismatch kernel. We compute the kernels efficiently using a trie data structure and relate our new kernels to the recently described transducer formalism. In protein classification experiments on two benchmark SCOP data sets, we show that our new faster kernels achieve SVM classification performance comparable to the mismatch kernel and the Fisher kernel derived from profile hidden Markov models, and we investigate the dependence of the kernels on parameter choice.

Keywords: kernel methods, string kernels, computational biology

1. Introduction

The original work on string kernels – kernel functions defined on the set of sequences from an alphabet Σ rather than on a vector space (Cristianini and Shawe-Taylor, 2000) – came from the field of computational biology and was motivated by algorithms for aligning DNA and protein sequences. Pairwise alignment algorithms, in particular the Smith-Waterman algorithm for optimal local alignment and the Needleman-Wunsch algorithm for optimal global alignment (Waterman et al., 1991), model the evolutionary process of mutations – insertions, deletions, and residue substitutions relative to an ancestral sequence – and give natural sequence similarity scores related to evolutionary distance. However, standard pairwise alignment scores do not represent valid kernels (Vert et al., 2004), and the first string kernels to be defined – dynamic alignment kernels based on pair hidden Markov models by Watkins (1999) and convolution kernels introduced by Haussler (1999) – had to develop new technical machinery to translate ideas from alignment algorithms into a kernel framework. More recently, there has also been interest in the development of string kernels for use with

support vector machine classifiers (SVMs) and other kernel methods in fields outside computational biology, such as text processing and speech recognition. For example, the gappy *n*-gram kernel developed by Lodhi et al. (2002) implemented a dynamic alignment kernel for text classification. A practical disadvantage of all these string kernels is their computational expense. In general, these kernels rely on dynamic programming algorithms for which the computation of each kernel value $K(x,y)$ is quadratic in the length of the input sequences x and y, that is, $O(|x||y|)$ with constant factor that depends on the parameters of the kernel.

The recently presented k-spectrum (gap-free k-gram) kernel and the (k,m) mismatch kernel provide an alternative model for string kernels for biological sequences and were designed, in particular, for the application of SVM protein classification. These kernels use counts of common occurrences of short k-length subsequences, called k-mers, rather than notions of pairwise sequence alignment, as the basis for sequence comparison. The k-mer idea still captures a biologically-motivated model of sequence similarity, in that sequences that diverge through evolution are still likely to contain short subsequences that match or almost match. Leslie et al. (2002a) introduced a linear time $(O(k(|x|+|y|))$ implementation of the k-spectrum kernel, using exact matches of k-mer patterns only, based on a trie data structure. Later, the (k,m)-mismatch kernel (Leslie et al., 2002b) extended the k-mer based kernel framework and achieved improved performance on the protein classification task by incorporating the biologically important notion of character mismatches (residue substitutions). Using a mismatch tree data structure, the complexity of the kernel calculation was shown to be $O(c_K(|x|+|y|))$, with $c_K = k^{m+1}|\Sigma|^m$ for k-grams with up to m mismatches from alphabet Σ. A different extension of the k-mer framework was presented by Vishwanathan and Smola (2002), who computed the weighted sum of exact-matching k-spectrum kernels for different k by using suffix trees and suffix links, allowing elimination of the constant factor in the spectrum kernel for a compute time of $O(|x|+|y|)$.

In this paper, we extend the k-mer based kernel framework in new ways by presenting several novel families of string kernels for use with SVMs for classification of protein sequence data. These kernels – restricted gappy kernels, substitution kernels, and wildcard kernels – are again based on feature spaces indexed by k-length subsequences from the string alphabet Σ (or the alphabet augmented by a wildcard character) and use biologically-inspired models of inexact matching. Thus the new kernels are closely related both to the (k,m)-mismatch kernel and the gappy k-gram string kernels used in text classification. However, for all kernels we define here, the kernel value $K(x,y)$ can be computed in $O(c_K(|x|+|y|))$ time, where the constant c_K depends on the parameters of the kernel but is independent of the size $|\Sigma|$ of the alphabet. Our efficient computation uses a recursive function based on a trie data structure and is linear in the length of the sequences, like the mismatch kernel, but we improve upon the parameter-dependent constant; a similar trie-based sequence search strategy has been used, for example, in work of Sagot (1998) for motif discovery. The restricted gappy kernels we present here can be seen as a fast approximation of the gappy k-gram kernel of Lodhi et al. (2002), where by using our k-mer based computation, we avoid dynamic programming and the resultant quadratic compute time; we note that the gappy k-gram kernel can also be seen as a special case of a dynamic alignment kernel (Watkins, 1999), giving a link between this work and some of the kernels we define. Cortes et al. (2002) have recently presented a transducer formalism for defining rational string kernels, and all the k-mer based kernels can be naturally described in this framework. We relate our new kernels to the transducer formalism and give transducers corresponding to our newer kernels. We note that Cortes et al. (2002) also describe the original convolution kernels of Haussler (1999) within their framework, suggesting that the transducer formalism is a

natural unifying framework for describing many string kernels in the literature. However, the complexity for kernel computation using the standard rational kernel algorithm is $O(c_T|x||y|)$, where c_T is a constant that depends on the transducer, leading again to quadratic rather than linear dependence on sequence length.

Finally, we report protein classification experiments on two benchmark data sets based on the SCOP database (Murzin et al., 1995), where we show that our new faster kernels achieve SVM classification performance comparable to the mismatch kernel and the Fisher kernel derived from profile hidden Markov models. We also use kernel alignment scores (Cristianini et al., 2001) to investigate how different the various inexact matching models are from each other, and to what extent they depend on kernel parameters. Moreover, we show that by using linear combinations of different kernels, we can improve performance over the best individual kernel.

The current paper is an extended version of the original paper presenting these inexact string matching kernels (Leslie and Kuang, 2003). We have added the second set of SCOP experiments to allow more investigation of the dependence of SVM performance on kernel parameter choices. We have also included the kernel alignment results to explore differences between kernels and parameter choices and the advantage of combining these kernels.

2. Definitions of Feature Maps and String Kernels

Below, we review the definition of mismatch kernels (Leslie et al., 2002b) and present three new families of string kernels: restricted gappy kernels, substitution kernels, and wildcard kernels.

In each case, the kernel is defined via an explicit feature map from the space of all finite sequences from an alphabet Σ to a vector space indexed by the set of k-length subsequences from Σ or, in the case of wildcard kernels, Σ augmented by a wildcard character. For protein sequences, Σ is the alphabet of $|\Sigma| = 20$ amino acids. We refer to a k-length contiguous subsequence occurring in an input sequence as an instance k-mer (also called a k-gram in the literature). The mismatch kernel feature map obtains inexact matching of instance k-mers from the input sequence to k-mer features by allowing a restricted number of mismatches; the new kernels achieve inexact matching by allowing a restricted number of gaps, by enforcing a probabilistic threshold on character substitutions, or by permitting a restricted number of matches to wildcard characters. These models for inexact matching have all been used in the computational biology literature in other contexts, in particular for sequence pattern discovery in DNA and protein sequences (Sagot, 1998) and probabilistic models for sequence evolution (Henikoff and Henikoff, 1992, Schwartz and Dayhoff, 1978, Altschul et al., 1990).

2.1 Spectrum and Mismatch Kernels

In previous work, we defined the (k,m)-mismatch kernel via a feature map $\Phi^{\text{Mismatch}}_{(k,m)}$ to the $|\Sigma|^k$-dimensional vector space indexed by the set of k-mers from Σ. For a fixed k-mer $\alpha = a_1 a_2 \ldots a_k$, with each a_i a character in Σ, the (k,m)-neighborhood generated by α is the set of all k-length sequences β from Σ that differ from α by at most m mismatches. We denote this set by $N_{(k,m)}(\alpha)$. For a k-mer α, the feature map is defined as

$$\Phi^{\text{Mismatch}}_{(k,m)}(\alpha) = (\phi_\beta(\alpha))_{\beta \in \Sigma^k}$$

where $\phi_\beta(\alpha) = 1$ if β belongs to $N_{(k,m)}(\alpha)$, and $\phi_\beta(\alpha) = 0$ otherwise. For a sequence x of any length, we extend the map additively by summing the feature vectors for all the k-mers in x:

$$\Phi^{\text{Mismatch}}_{(k,m)}(x) = \sum_{k\text{-mers } \alpha \text{ in } x} \Phi^{\text{Mismatch}}_{(k,m)}(\alpha).$$

Each instance of a k-mer contributes to all coordinates in its mismatch neighborhood, and the β-coordinate of $\Phi^{\text{Mismatch}}_{(k,m)}(x)$ is just a count of all instances of the k-mer β occurring with up to m mismatches in x. The (k,m)-mismatch kernel $K_{(k,m)}$ is then given by the inner product of feature vectors:

$$K^{\text{Mismatch}}_{(k,m)}(x,y) = \langle \Phi^{\text{Mismatch}}_{(k,m)}(x), \Phi^{\text{Mismatch}}_{(k,m)}(y) \rangle.$$

For $m = 0$, we obtain the k-spectrum (Leslie et al., 2002a) or k-gram kernel (Lodhi et al., 2002).

2.2 Restricted Gappy Kernels

For the (g,k)-gappy string kernel, $g \geq k$, we use the same $|\Sigma|^k$-dimensional feature space, indexed by the set of k-mers from Σ, but we define our feature map based on gappy matches of g-mers to k-mer features. For a fixed g-mer $\alpha = a_1 a_2 \ldots a_g$(each $a_i \in \Sigma$), let $G_{(g,k)}(\alpha)$ be the set of all the k-length subsequences occurring in α (with up to $g-k$ gaps). Then we define the gappy feature map on α as

$$\Phi^{\text{Gap}}_{(g,k)}(\alpha) = (\phi_\beta(\alpha))_{\beta \in \Sigma^k},$$

where $\phi_\beta(\alpha) = 1$ if β belongs to $G_{(g,k)}(\alpha)$, and $\phi_\beta(\alpha) = 0$ otherwise. In other words, each instance g-mer contributes to the set of k-mer features that occur (in at least one way) as subsequences with up to $g-k$ gaps in the g-mer. Now we extend the feature map to arbitrary finite sequences x by summing the feature vectors for all the g-mers in x:

$$\Phi^{\text{Gap}}_{(g,k)}(x) = \sum_{\text{g-mers } \alpha \in x} \Phi^{\text{Gap}}_{g,k}(\alpha).$$

The kernel $K^{\text{Gap}}_{(g,k)}(x,y)$ is defined as before by taking the inner product of feature vectors for x and y.

Alternatively, given an instance g-mer, we may wish to count the number of occurrences of each k-length subsequence and weight each occurrence by the number of gaps. Following (Lodhi et al., 2002), we can define for g-mer α and k-mer feature $\beta = b_1 b_2 \ldots b_k$ the weighting

$$\phi^\lambda_\beta(\alpha) = \frac{1}{\lambda^k} \sum_{\substack{1 \leq i_1 < i_2 < \ldots < i_k \leq g \\ a_{i_j} = b_j \text{ for } j = 1 \ldots k}} \lambda^{i_k - i_1 + 1},$$

where the multiplicative factor satisfies $0 < \lambda \leq 1$. We can then obtain a weighted version of the gappy kernel $K^{\text{Weighted Gap}}_{(g,k,\lambda)}$ from the feature map:

$$\Phi^{\text{Weighted Gap}}_{(g,k,\lambda)}(x) = \sum_{\text{g-mers } \alpha \in x} (\phi^\lambda_\beta(\alpha))_{\beta \in \Sigma^k}.$$

Here, the weighting $\dfrac{\lambda^{i_k - i_1 + 1}}{\lambda^k}$ penalizes a gappy occurrence of a k-mer by a factor λ raised to the number of internal gaps. This feature map is related to the gappy k-gram kernel defined by Lodhi

et al. (2002) but enforces the following restriction: here, only those k-character subsequences that occur with at most $g-k$ gaps, rather than all gappy occurrences, contribute to the corresponding k-mer feature. When restricted to input sequences of length g, our feature map coincides with that of the usual gappy k-gram kernel. Note, however, that for our kernel, a gappy k-mer instance (occurring with at most $g-k$ gaps) is counted in all (overlapping) g-mers that contain it, whereas in Lodhi et al. (2002), a gappy k-mer instance is only counted once. If we wish to approximate the gappy k-gram kernel, we can define a small variation of our restricted gappy kernel where one only counts a gappy k-mer instance if its first character occurs in the first position of a g-mer window. That is, the modified feature map is defined on each g-mer α by coordinate functions

$$\widetilde{\phi}^{\lambda}_{\beta}(\alpha) = \frac{1}{\lambda^k} \sum_{\substack{1=i_1<i_2<\ldots<i_k\leq g \\ a_{i_j}=b_j \text{ for } j=1\ldots k}} \lambda^{i_k-i_1+1},$$

$0<\lambda\leq 1$, and is extended to longer sequences by adding feature vectors for g-mers. This modified feature map now gives a "truncation" of the usual gappy k-gram kernel.

In Section 3, we show that our restricted gappy kernel has $O(c(g,k)(|x|+|y|))$ computation time, where constant $c(g,k)$ depends on size of g and k, while the original gappy k-gram kernel has complexity $O(k(|x||y|))$. Note in particular that we do not compute the standard gappy k-gram kernel on every pair of g-grams from x and y, which would necessarily be quadratic in sequence length since there are $O(|x||y|)$ such pairs. We will see that for reasonable choices of g and k, we obtain much faster computation time, while in experimental results reported in Section 5, we still obtain good classification performance.

2.3 Substitution Kernels

The substitution kernel is again similar to the mismatch kernel, except that we replace the combinatorial definition of a mismatch neighborhood with a similarity neighborhood based on a probabilistic model of character substitutions. In computational biology, it is standard to compute pairwise alignment scores for protein sequences using a substitution matrix (Henikoff and Henikoff, 1992, Schwartz and Dayhoff, 1978, Altschul et al., 1990) that gives pairwise scores $s(a,b)$ derived from estimated evolutionary substitution probabilities. In one scoring system (Schwartz and Dayhoff, 1978), the scores $s(a,b)$ are based on estimates of conditional substitution probabilities $P(a|b)=p(a,b)/q(b)$, where $p(a,b)$ is the probability that a and b co-occur in an alignment of closely related proteins, $q(a)$ is the background frequency of amino acid a, and $P(a|b)$ represents the probability of a mutation into a during fixed evolutionary time interval given that the ancestor amino acid was b. We define the mutation neighborhood $M_{(k,\sigma)}(\alpha)$ of a k-mer $\alpha=a_1a_2\ldots a_k$ as follows:

$$M_{(k,\sigma)}(\alpha) = \{\beta=b_1b_2\ldots b_k\in\Sigma^k : -\sum_{i}^{k}\log P(a_i|b_i)<\sigma\}.$$

Mathematically, we can define $\sigma=\sigma(N)$ such that $\max_{\alpha\in\Sigma^k}|M_{(k,\sigma)}(\alpha)|<N$, so we have theoretical control over the maximum size of the mutation neighborhoods. In practice, choosing σ to allow an appropriate amount of mutation while restricting neighborhood size may require experimentation and cross-validation.

Now we define the substitution feature map analogously to the mismatch feature map:

$$\Phi^{\text{Sub}}_{(k,\sigma)}(x) = \sum_{k\text{-mers } \alpha \text{ in } x} (\phi_\beta(\alpha))_{\beta\in\Sigma^k},$$

where $\phi_\beta(\alpha) = 1$ if β belongs to the mutation neighborhood $M_{(k,\sigma)}(\alpha)$, and $\phi_\beta(\alpha) = 0$ otherwise.

2.4 Wildcard Kernels

Finally, we can augment the alphabet Σ with a wildcard character denoted by $*$, and we map to a feature space indexed by the set $\mathcal{W}$ of k-length subsequences from $\Sigma \cup \{*\}$ having at most m occurrences of the character $*$. The feature space has dimension $\sum_{i=0}^{m} \binom{k}{i} |\Sigma|^{k-i}$.

A k-mer α matches a subsequence β in $\mathcal{W}$ if all non-wildcard entries of β are equal to the corresponding entries of α (wildcards match all characters). The wildcard feature map is given by

$$\Phi^{\text{Wildcard}}_{(k,m,\lambda)}(x) = \sum_{k\text{-mers } \alpha \text{ in } x} (\phi_\beta(\alpha))_{\beta \in \mathcal{W}},$$

where $\phi_\beta(\alpha) = \lambda^j$ if α matches pattern β containing j wildcard characters, $\phi_\beta(\alpha) = 0$ if α does not match β, and $0 < \lambda \leq 1$.

Other variations of the wildcard idea, including specialized weightings and use of groupings of related characters, are described by Eskin et al. (2003).

3. Efficient Computation

All the k-mer based kernels we define above can be efficiently computed using a trie (retrieval tree) data structure, similar to the mismatch tree approach previously presented (Leslie et al., 2002b). In this framework, k-mer features correspond to paths from the root to the leaf nodes of the tree, and the data structure is used to organize a traversal of all the inexact matching instance patterns in the data that contribute to each k-mer feature count. We will describe the computation of the gappy kernel in most detail, since the other kernels are easier adaptations of the mismatch kernel computation. For simplicity, we explain how to compute a single kernel value $K(x,y)$ for a pair of input sequences; computation of the full kernel matrix in one traversal of the data structure is a straightforward extension.

3.1 (g,k)-Gappy Kernel Computation

For the (g,k)-gappy kernel, we represent our feature space as a rooted tree of depth k where each internal node has $|\Sigma|$ branches and each branch is labeled with a symbol from Σ. In this depth k trie, each leaf node represents a fixed k-mer in feature space by concatenating the branch symbols along the path from root to leaf and each internal node represents the prefix for those for the set of k-mer features in the subtree below it.

Using a depth-first traversal of this tree, we maintain at each node that we visit a set of pointers to all g-mer instances in the input sequences that contain a subsequence (with gaps) that matches the current prefix pattern; we also store, for each g-mer instance, an index pointing to the last position we have seen so far in the g-mer. At the root, we store pointers to all g-mer instances, and for each instance, the stored index is 0, indicating that we have not yet seen any characters in the g-mer. As we pass from a parent node to a child node along a branch labeled with symbol a, we process each of parent's instances by scanning ahead to find the next occurrence of symbol a in each g-mer. If such a character exists, we pass the g-mer to the child node along with its updated index; otherwise, we drop the instance and do not pass it to the child. Thus at each node of depth d, we

have effectively performed a greedy gapped alignment of g-mers from the input sequences to the current d-length prefix, allowing insertion of up to $g-k$ gaps into the prefix sequence to obtain each alignment. When we encounter a node with an empty list of pointers (no valid occurrences of the current prefix), we do not need to search below it in the tree; in fact, unless there is a valid g-mer instance from each of x and y, we do not have to process the subtree. When we reach a leaf node, we sum the contributions of all instances occurring in each source sequence to obtain feature values for x and y corresponding to the current k-mer, and we update the kernel by adding the product of these feature values. Since we are performing a depth-first traversal, we can accomplish the algorithm with a recursive function and do not have to store the full trie in memory. Figure 1 shows expansion down a path during the recursive traversal.

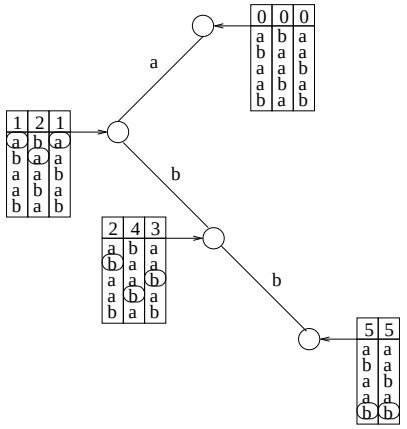


Figure 1: **Trie traversal for gappy kernel.** Expansion along a path from root to leaf during traveral of the trie for the $(5,3)$-gappy kernel, showing only the instance 5-mers for a single sequence $x = abaabab$. Each node stores its valid 5-mer instances and the index to the last match for each instance. Instances at the leaf node contribute to the kernel for 3-mer feature *abb*.

The computation at the leaf node depends on which version of the gappy kernel one uses. For the unweighted feature map, we obtain the feature values of x and y corresponding to the current k-mer by counting the g-mer instances at the leaf coming from x and from y, respectively; the product of these counts gives the contribution to the kernel for this k-mer feature. For the λ-weighted gappy feature map, we need a count of all alignments of each valid g-mer instance against the k-mer feature allowing up to $g-k$ gaps. This can be computed with a simple dynamic programming routine (similar to the Needleman-Wunsch algorithm), where we sum over a restricted set of paths, as shown in Figure 2. The complexity is $O(k(g-k))$, since we fill a restricted trellis of $(k+1)(g-k+1)$ squares. Note that when we align a subsequence $b_{i_1}b_{i_2}\ldots b_{i_k}$ against a k-mer $a_1a_2\ldots a_k$, we only penalize interior gaps corresponding to non-consecutive indices in $1 \leq i_1 < i_2 \ldots < i_k \leq g$. Therefore, the multiplicative gap cost is 1 in the zeroth and last rows of the trellis and λ in the other rows.

In order to determine the worst case complexity of the kernel computation, we estimate the traversal time – which can be bounded by the total number of g-mer instances that are processed

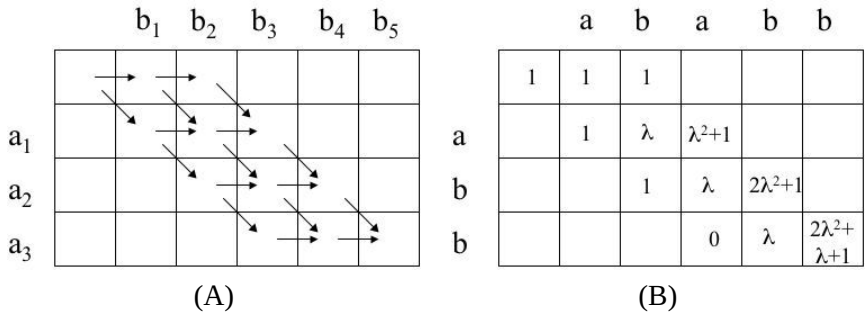


Figure 2: **Dynamic programming at the leaf node.** The trellis in (A) shows the restricted paths for aligning a g-mer against a k-mer, with insertion of up to $g-k$ gaps in the k-mer, for $g=5$ and $k=3$. The basic recursion for summing path weights is $S(i,j) = m(a_i,b_j)S(i-1,j-1) + g(i)S(i,j-1)$, where $m(a,b) = 1$ if a and b match, 0 if they are different, and the gap penalty $g(i) = 1$ for $i = 0,k$ and $g(i) = \lambda$ for other rows. That is, except for the top and bottom rows, every time we move to the right, we introduce an additional internal gap and incur a multiplicative penalty of λ; when we move diagonally, we see whether the corresponding characters match or not. Trellis (B) shows the example of aligning *ababb* against 3-mer *abb*. An explanation of how to interpret trellis diagrams for dynamic programming can be found in Durbin et al. (1998).

at the leaf nodes multiplied by the maximum number of times an instance is operated on as it is passed from root to leaf – plus the processing time at the leaf nodes need to compute the kernel update. Each g-mer instance in the input data can contribute to $\binom{g}{k}$ k-mer features, which we can write as $O(g^{g-k})$ if $g-k$ is smaller than k and $O(g^k)$ otherwise. For simplicity, we assume the more typical former case, and we let the reader make simple adjustments in the latter case. Therefore, at most $O(g^{g-k}(|x|+|y|)$ g-mer instances are processed at leaf nodes in the traversal. Since we iterate through at most g positions of each g-mer instance as we pass from root to leaf, the traversal time is $O(g^{g-k+1}(|x|+|y|))$. The total processing time at leaf nodes is $O(g^{g-k}(|x|+|y|))$ for the unweighted gappy kernel and $O(k(g-k)g^{g-k}(|x|+|y|))$ for the weighted gappy kernel. Therefore, in both cases, we have total complexity of the form $O(c(g,k)(|x|+|y|))$, with $c(g,k) = O((g-k)g^{g-k+1})$ for the more expensive kernel. Further discussion of the complexity argument and pseudocode for the algorithm can be found in Shawe-Taylor and Cristianini (2004).

Note that with the definition of the gappy feature maps given above, a gappy k-character subsequence occuring with $c \leq g-k$ gaps is counted in each of the $g-(k+c)+1$ g-length windows that contain it. To obtain feature maps that count a gappy k-character subsequence only once, we can make minor variations to the algorithm by requiring that the first character of a gappy k-mer occurs in the first position of the g-length window in order to contribute to the corresponding k-mer feature.

3.2 (k,σ)-Substitution Kernel Computation

For the substitution kernel, computation is very similar to the mismatch kernel algorithm. We use a depth k trie to represent the feature space. We store, at each depth d node that we visit, a set of pointers to all k-mer instances α in the input data whose d-length prefixes have current mutation score $-\sum_{i=1}^{d}\log P(a_i|b_i) < \sigma$ of the current prefix pattern $b_1 b_2 \ldots b_d$, and we store the current mutation score for each k-mer instance. As we pass from a parent node at depth d to a child node at depth $d+1$ along a branch labeled with symbol b, we process each k-mer α by adding $-\log P(a_{d+1}|b)$ to the mutation score and pass it to the child if and only if the score is still less than σ. As before, we update the kernel at the leaf node by computing the contribution of the corresponding k-mer feature.

The number of leaf nodes visited in the traversal is $O(N_\sigma(|x|+|y|))$, where the constant is the maximum mutation neighborhood size, $N_\sigma = \max_{\alpha\in\Sigma^k}|M_{(k,\sigma)}|$. We can choose σ sufficiently small to get any desired bound on N_σ, but it is difficult to estimate how to set the parameters to obtain good SVM classification performance except by empirical results. Total complexity for the kernel value computation is $O(kN_\sigma(|x|+|y|))$.

3.3 (k,m)-Wildcard Kernel Computation

Computation of the wildcard kernel is again very similar to the mismatch kernel algorithm. We use a depth k trie with branches labeled by characters in $\Sigma\cup\{*\}$, and we prune (do not traverse) subtrees corresponding to prefix patterns with greater than m wildcard characters. At each node of depth d, we maintain pointers to all k-mers instances in the input sequences whose d-length prefixes match the current d-length prefix pattern (with wildcards) represented by the path down from the root.

Each k-mer instance in the data matches at most $\sum_{i=0}^{m}\binom{k}{i} = O(k^m)$ k-length patterns having up to m wildcards. Thus the number of leaf nodes visited is in the traversal is $O(k^m(|x|+|y|))$, and total complexity for the kernel value computation is $O(k^{m+1}(|x|+|y|))$.

3.4 Comparison with Mismatch Kernel Complexity

For the (k,m) mismatch kernel, the size of the mismatch neighborhood of an instance k-mer is $O(k^m|\Sigma|^m)$, so total kernel value computation is $O(k^{m+1}|\Sigma|^m(|x|+|y|))$. All the other kernels presented here have running time $O(c_K(|x|+|y|))$, where constant c_K depends on the parameters of the kernel but not on the size of the alphabet Σ. Therefore, we have improved constant term for larger alphabets (such as the alphabet of 20 amino acids). In Section 5, we show that these new, faster kernels have performance comparable to the mismatch kernel in protein classification experiments.

4. Transducer Representation

Cortes et al. (2002) recently showed that many known string kernels can be associated with and constructed from weighted finite state transducers with input alphabet Σ. We briefly outline their transducer formalism and give transducers for some of our newly defined kernels. For simplicity, we only describe transducers over the probability semiring $\mathbb{R}_+ = [0,\infty)$, with regular addition and multiplication.

Following the development in Cortes et al. (2002), a weighted finite state transducer over $\mathbb{R}_+$ is defined by a finite input alphabet Σ, a finite output alphabet Δ, a finite set of states Q, a set of input

states $I \subset Q$, a set of output states $F \subset Q$, a finite set of transitions $E \subset Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times \mathbb{R}_+ \times Q$, an initial weight function $\lambda : I \to \mathbb{R}_+$, and a final weight function $\rho : F \to \mathbb{R}_+$. Here, the symbol ε represents the empty string. The transducer can be represented by a weighted directed graph with nodes indexed by Q and each transition $e \in E$ corresponding to a directed edge from its origin state $p[e]$ to its destination state $n[e]$ and labeled by the input symbol $i[e]$ it accepts, the output symbol $o[e]$ it emits, and the weight $w[e]$ it assigns. We write the label as $i[e] : o[e]/w[e]$ (abbreviated as $i[e] : o[e]$ if the weight is 1).

For a path $\pi = e_1 e_2 \dots e_k$ of consecutive transitions (directed path in graph), the weight for the path is $w[\pi] = w[e_1]w[e_2]\dots w[e_k]$, and we denote $p[\pi] = p[e_1]$ and $n[\pi] = n[e_k]$. We write $\Sigma^* = \cup_{k \geq 0} \Sigma^k$ for the set of all strings over Σ. For an input string $x \in \Sigma^*$ and output string $z \in \Delta^*$, we denote by $P(I, x, z, F)$ the set of paths from initial states I to final states F that accept string x and emit string z. A transducer T is called regulated if for any pair of input and output strings (x, z), the output weight $[[T]](x, z)$ that T assigns to the pair is well-defined. The output weight is given by:

$$[[T]](x,z) = \sum_{\pi \in P(I,x,z,F)} \lambda(p[\pi]) w[\pi] \rho(n[\pi])$$

A key observation from Cortes et al. (2002) is that there is a general method for defining a string kernel from a weighted transducer T. Let $\Psi : \mathbb{R}_+ \to \mathbb{R}$ be a semiring morphism (for us, it will simply be inclusion), and denote by T^{-1} the transducer obtained from T by transposing the input and output labels of each transition. Then if the composed transducer $S = T \circ T^{-1}$ is regulated, one obtains a rational string kernel for alphabet Σ via

$$K(x,y) = \Psi([[S]](x,y)) = \sum_z \Psi([[T]](x,z)) \Psi([[T]](y,z))$$

where the sum is over all strings $z \in \Delta^*$ (where Δ is the output alphabet for T) or equivalently, over all output strings that can be emitted by T. Therefore, we can think of T as defining a feature map indexed by all possible output strings $z \in \Delta^*$ for T.

Using this construction, Cortes et al. showed that the k-gram counter transducer T_k corresponds to the k-gram or k-spectrum kernel, and the gappy k-gram counter transducer $T_{k,\lambda}$ gives the unrestricted gappy k-gram kernel from Lodhi et al. (2002). Figure 3(a) shows the diagram of the 3-gram transducer T_3, and Figure 3(b) gives the gappy 3-gram transducer $T_{3,\lambda}$. Our (g,k,λ)-gappy kernel $K^{\text{Weighted Gap}}_{(g,k,\lambda)}$ can be obtained from the composed transducer $T = T_{k,\lambda} \circ T_g$ using the $T \circ T^{-1}$ construction. (In all our examples, we use $\lambda(s) = 1$ for every initial state s and $\rho(t) = 1$ for every final state t.)

For the (k,m)-wildcard kernel, we set the output alphabet to be $\Delta = \Sigma \cup \{*\}$ and define a transducer with $m+1$ final states, as indicated in the figure. The $m+1$ final states correspond to destinations of paths that emit k-grams with 0, 1, …, m wildcard characters, respectively. The $(3,1)$-wildcard transducer is shown in Figure 4.

The (k,σ)-substitution kernel does not appear to fall exactly into this framework, though if we threshold individual substitution probabilities independently rather than threshold the product probability over all positions in the k-mer, we can define a transducer that generates a similar kernel. Starting with the k-gram transducer, we add additional transitions (between "consecutive" states of the k-gram) of the form $a : b$ for those pairs of symbols with $-\log P(a|b) < \sigma_o$. Now there will be a (unique) path in the transducer that accepts k-mer $\alpha = a_1 a_2 \dots a_k$ and emits $\beta = b_1 b_2 \dots b_k$ if and only if every substitution satisfies $-\log P(a_i|b_i) < \sigma_o$.

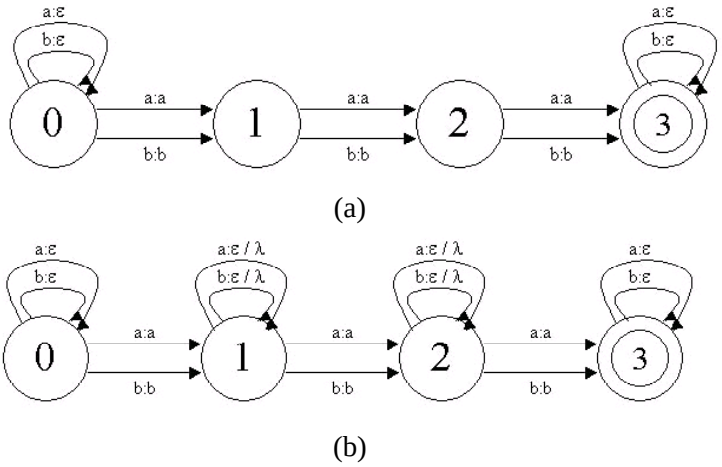


Figure 3: **The k-gram and gappy k-gram transducers.** The diagrams show the 3-gram transducer (a) and the gappy 3-gram transducer (b) for a two-letter alphabet. Here, the edge label $a : \varepsilon : \lambda$, for example, means "accept symbol a, output the empty symbol ε, multiply the weight by λ".

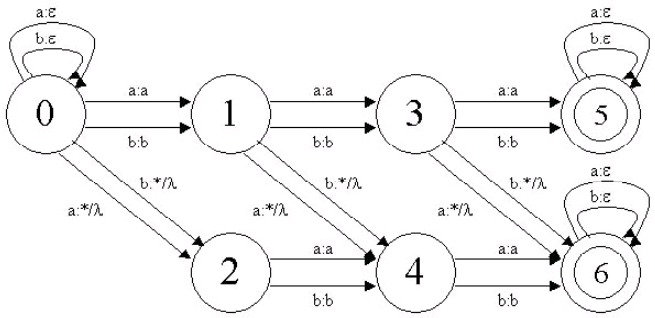


Figure 4: **The (k,m)-wildcard transducer.** The diagram shows the $(3,1)$-wildcard transducer for a two-letter alphabet.

5. Experiments

We tested all the new string kernels with SVM classifiers on two benchmark data sets (Jaakkola et al., 1999, Weston et al., 2003), both designed for the remote protein homology detection problem, in order to compare to results with the mismatch kernel reported previously (Leslie et al., 2002b) and other recent kernel representations for protein sequence data. We also present results to explore how parameter choices for the new kernels affect SVM classification performance. The benchmarks are based on different versions of the SCOP database (Murzin et al., 1995), an expert-curated database of protein domains with known 3D structure, organized hierarchically into folds, superfamilies, and

families. Protein domain sequences belonging to different families but the same superfamily are considered to be remote homologs in SCOP. In these experiments, remote homology is simulated by holding out all members of a target SCOP family from a given superfamily as a test set, while examples chosen from the remaining families in the same superfamily form the positive training set. The negative test and training examples are chosen from disjoint sets of folds outside the target family's fold, so that negative test and negative training sets are unrelated to each other and to the positive examples. More details of the experimental set-up can be found in Jaakkola et al. (1999).

While in principle, we can define and test inexact matching string kernels for a wide range of parameters, in practice, only a small parameter range is biologically motivated for use in the remote protein homology detection problem. For the exact matching k-spectrum kernel (Leslie et al., 2002a), the only interesting parameter choices are $k = 3$ and $k = 4$, since exact occurrences of k-mers of length $k \geq 5$ in remotely homologous proteins are so rare that the spectrum kernel would mostly be 0 off the diagonal. By incorporating inexact matching such as mismatches or gaps, we can use slightly longer subsequence instances and allow a few mismatches or gaps; however, using very long subsequences, or allowing a great amount of mutation of subsequence instances in our inexact matching scheme, would not capture biologically realistic sequence similarity. For example, for the gappy kernel, we expect that length $(g,k) = (6,4)$ – 4-mers allowing up to 2 gaps – would be a useful parameter choice, while allowing many more gaps and hence a longer g-length window would be less useful. We test a range of parameter choices that seem reasonable in the experiments below.

In the first and larger SCOP benchmark data set, based on SCOP version 1.37, we compare to the Fisher kernel of Jaakkola et al. (1999) in addition to our previous mismatch kernel. In the Fisher kernel method, the feature vectors are derived from profile HMMs trained on the positive training examples. The feature vector for sequence x is the gradient of the log likelihood function $\log P(x|\theta)$ defined by the model and evaluated at the maximum likelihood estimate for model parameters: $\Phi(x) = \nabla_\theta \log P(x|\theta)|_{\theta=\theta_0}$. The Fisher kernel was the best performing method on this data set prior to the mismatch-SVM approach, whose performance is as good as Fisher-SVM and better than all other standard methods tried (Leslie et al., 2002b). We note that in this data set, additional positive training sequences were pulled in from the non-redundant protein sequence database using an iterative training method for the profile HMMs. The presence of these additional "domain homologs" makes the learning task easier for all methods.

We also include a second set of experiments to further investigate the dependence of SVM performance on parameter choices for the new kernels. This second data set is based on SCOP version 1.59 and contains only sequences from the SCOP database – no domain homologs are added. The experiments are similar to those described by Liao and Noble (2002) but use a more recent version of SCOP. In this data set, the positive training sets are quite small, and the learning task is more difficult in this setting. In particular, there is not enough positive training data to train profile HMMs in these experiments, so we do not report Fisher kernel results (which are not competitive in this setting).

There is another successful feature representation for protein classification, the SVM-pairwise method presented in Liao and Noble (2002). Here one uses an empirical kernel map based on pairwise Smith-Waterman (Waterman et al., 1991) alignment scores

$$\Phi(x) = (d(x_1,x),\ldots,d(x_m,x)),$$

where x_i, $i = 1 \ldots m$, are the training sequences and $d(x_i, x)$ is the E-value for the alignment score between x and x_i. We have previously shown (Leslie et al., 2004) that the mismatch kernel used with an SVM classifier is competitive with SVM-pairwise on the SCOP benchmark presented in Liao and Noble (2002), so we do not repeat the SVM-pairwise experiments for the very similar benchmark here.

In both experiments, we normalized the kernel by $k(x,y) \leftarrow \frac{k(x,y)}{\sqrt{k(x,x)}\sqrt{k(y,y)}}$. All methods are evaluated using the receiver operating characteristic (ROC) score, which is the area under the receiver operating curve, which plots the rate of true positives as a function of the rate of false positives as the threshold for the classifier varies (Gribskov and Robinson, 1996). Perfect ranking of all positives above all negatives gives an ROC score of 1, while a random classifier has an expected score close to 0.5. We also use the ROC-50 score, which is the normalized area under the receiver operating curve up to the first 50 false positives; this score focuses on the top of the ranking of the test examples produced by the classifier and is more informative when there are very few positive examples in the test set. Use of ROC-50 scores (or other ROC-N scores) is the most standard way of evaluating performance of homology detection methods in bioinformatics (Gribskov and Robinson, 1996).

Finally, we use kernel alignment scores (Cristianini et al., 2001) on the second SCOP data set to investigate the empirical differences between the different inexact matching models for protein sequence data, and we investigate methods for combining kernels to improve SVM performance.

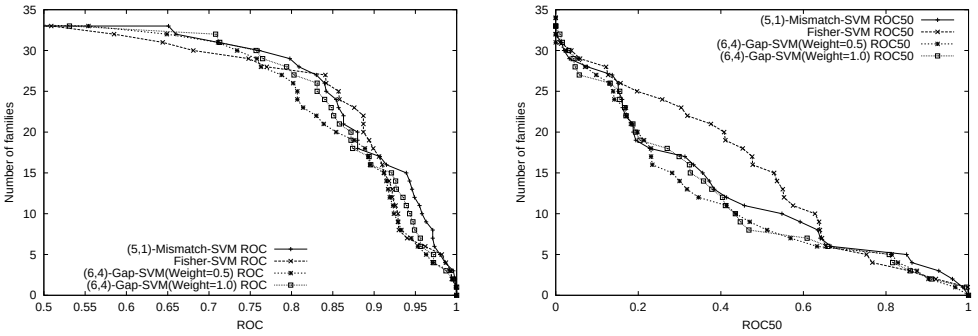
5.1 SCOP Experiments with Domain Homologs: Comparison with Fisher and Mismatch Kernels

We first present experimental results for the new kernels on the larger of the two SCOP data sets, the Fisher-SCOP benchmark introduced by Jaakkola et al. (1999) that contains domain homologs for additional positive training data, and compare SVM classification performance to both the mismatch kernel and the Fisher kernel.

We tested the (g,k)-gappy kernel with parameter choices $(g,k) = (6,4)$, $(7,4)$, $(8,5)$, $(8,6)$, and $(9,6)$. Among them $(g,k) = (6,4)$ yielded the best results, though other choices of parameters had quite similar performance (data not shown). We also tested the alternative weighted gappy kernel, where the contribution of an instance g-mer to a k-mer feature is a weighted sum of all the possible matches of the k-mer to subsequences in the g-mer with multiplicative gap penalty λ $(0 < \lambda \leq 1)$. We used gap penalty $\lambda = 1.0$ and $\lambda = 0.5$ with the $(6,4)$ weighted gappy kernel. We found that $\lambda = 0.5$ weighting slightly weakened performance (results not shown). In Figure 5, we see that unweighted and weighted ($\lambda = 1.0$) gappy kernels have comparable results to $(5,1)$-mismatch kernel and Fisher kernel.

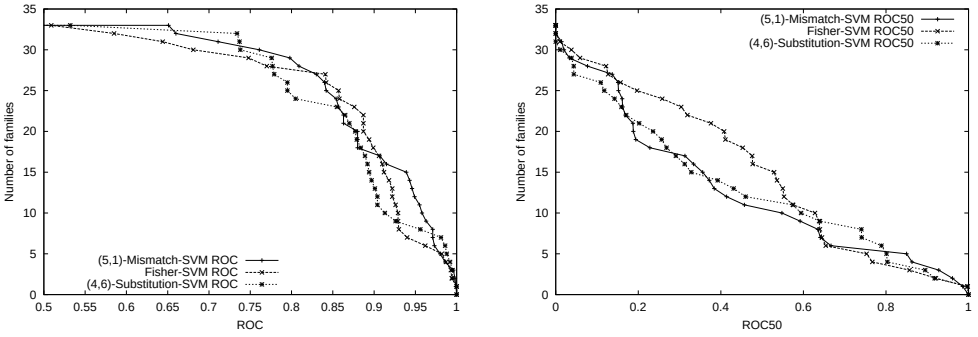
We tested the substitution kernels with $(k,\sigma) = (4,6.0)$. Here, $\sigma = 6.0$ was chosen so that the members of a mutation neighborhood of a particular 4-mer would typically have only one position with a substitution, and such substitutions would have fairly high probability. Therefore, the mutation neighborhoods were much smaller than, for example, $(4,1)$-mismatch neighborhoods. The results are shown in Figure 6. Again, the substitution kernel has comparable performance with mismatch-SVM and Fisher-SVM, though results are perhaps slightly weaker for more difficult test families.

In order to compare with the $(5,1)$-mismatch kernel, we tested wildcard kernels with parameters $(k,m,\lambda) = (5,1,1.0)$ and $(k,m,\lambda) = (5,1,0.5)$. Results are shown in Figure 7. The wildcard kernel with $\lambda = 1.0$ seems to perform as well or almost as well as the $(5,1)$-mismatch kernel and Fisher



(a) (b)

Figure 5: **Comparison of of Mismatch-SVM, Fisher-SVM and Gappy-SVM.** The graph plots the total number of families for which a given method exceeds an ROC score threshold (a) or ROC-50 score threshold (b).



(a) (b)

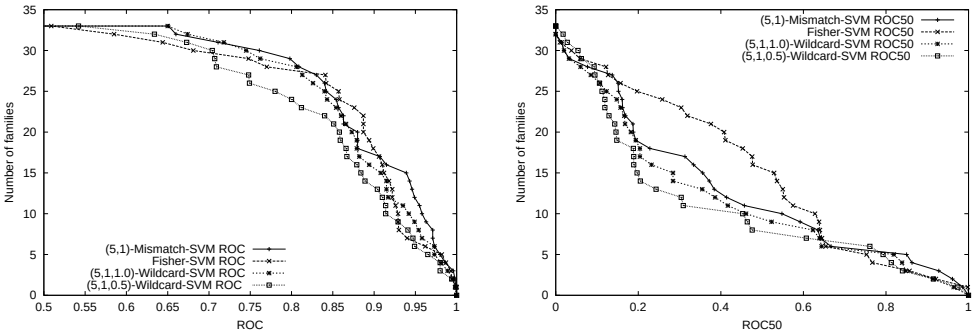
Figure 6: **Comparison of mismatch-SVM, Fisher-SVM and substitution-SVM.** The graph plots the total number of families for which a given method exceeds an ROC score threshold (a) or ROC-50 score threshold (b).

kernel, while enforcing a penalty on wildcard characters of $\lambda = 0.5$ seems to weaken performance somewhat.

If we compare results for the best-performing parameter choices that we tried from each kernel family – the $(5,1)$-mismatch kernel, the $(5,1,1.0)$-wildcard kernel, the $(6,4)$-gappy kernel with $\lambda = 1.0$, and the $(4,6.0)$-substitution kernel – then a signed ranked test with Bonferroni correction for multiple comparisons (Henikoff and Henikoff, 1992, Salzberg, 1997) and a p-value cut-off of 0.05 finds no significant differences between the four kernels, either on the basis of ROC or ROC-50 scores.

5.2 SCOP Experiments without Domain Homologs: Dependence on Parameters

In the second set of SCOP experiments, we take advantage of the smaller data set from Weston et al. (2003) to generate kernels corresponding to a wider range of parameter values, so that we

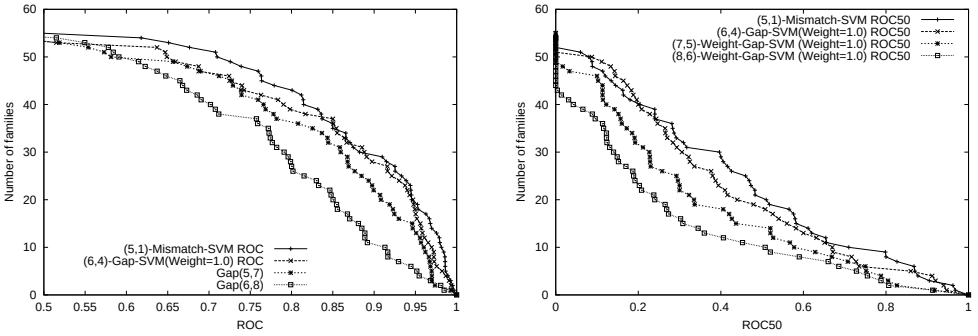


(a) (b)

Figure 7: **Comparison of mismatch-SVM, Fisher-SVM and wildcard-SVM.** The graph plots the total number of families for which a given method exceeds an ROC score threshold (a) or ROC-50 score threshold (b).

can explore how parameter choices affect SVM classification performance. We also use kernel alignment and kernel-target alignment scores (Cristianini et al., 2001) to investigate differences between different kernel models. Note that this data set contains no domain homologs, and thus the small amount of positive training data makes the experiments more difficult.

In experiments with the gappy kernel, we chose parameter values $(g,k) = (6,4)$, $(7,5)$ and $(8,6)$ and set the gap penalty to $\lambda = 1.0$, the preferred choice from the previous experiments. The choice $(g,k) = (6,4)$ still produced the best classification results, which were slightly but not significantly weaker than those of $(5,1)$-mismatch kernel. The results are shown in Figure 8. Performance deteriorates as larger values of the g parameter are chosen with the number of gaps held fixed.

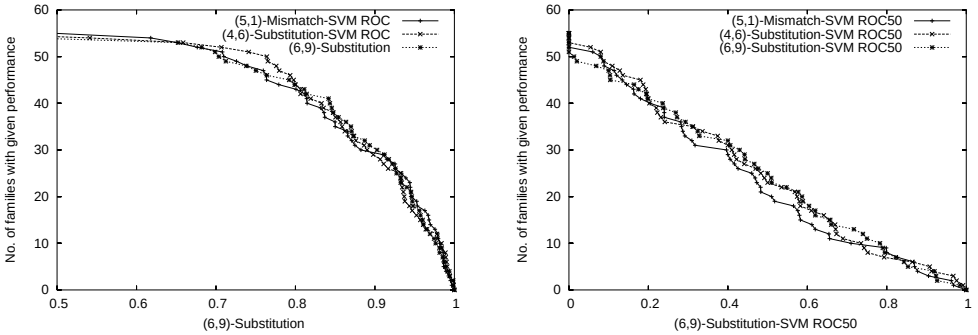


(a) (b)

Figure 8: **Dependence on parameters for the gappy kernel.** The graph plots the total number of families for which a given method exceeds an ROC (a) or ROC-50 (b) score threshold.

The substitution kernel was tested with parameter choices $(k,\sigma) = (4,6.0), (5,7.5)$ and $(6,9.0)$. All of these three kernels gave slightly stronger performance than the $(5,1)$-mismatch kernel, and results for the different parameter choices were remarkably similar, as shown in Figure 9. Thus, more so than for other inexact matching models, the substitution kernel performance seems stable

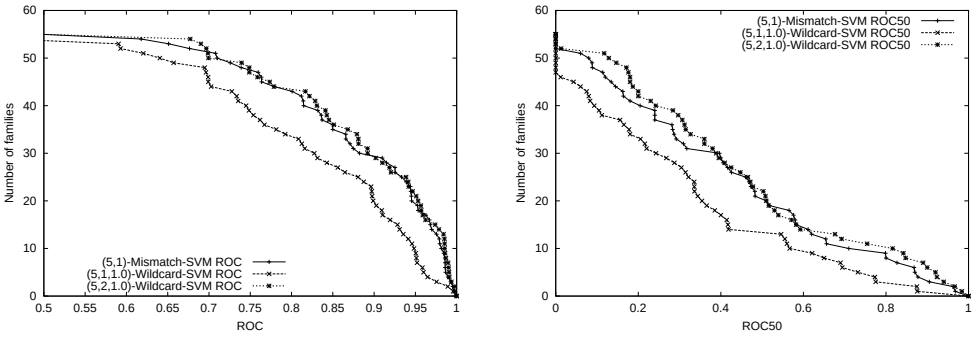
as we vary k while σ is adjusted additively; however, as we see below, the Gram matrices produced by these different choices of kernels are in fact quite different.



(a) (b)

Figure 9: **Dependence on parameters for substitution kernel.** The graph plots the total number of families for which a given method exceeds an ROC (a) or ROC-50 (b) score threshold. Results for three parameter choices give almost identical results.

We tested the wildcard kernel with $(k,m,\lambda) = (5,1,1.0)$ and $(5,2,1.0)$. We observed a significant improvement in performance when we allowed up to 2 wildcards instead of 1 with $k = 5$. The performance of $(5,2,1.0)$-wildcard kernel gave the best results among all kernel families and parameters that we tried, though several other kernel choices gave very similar performance. The results are shown in Figure 10. Intuitively, it is clear that allowing 1 mismatch is closer to permitting 2 wildcards than to permitting a single wildcard: two k-mers that are identical except in *two* positions have intersecting $(k,1)$-mismatch neighborhoods and hence their $(k,1)$-mismatch feature vectors have non-zero inner product; similarly, such a pair of k-mers have non-orthogonal $(k,2)$-wildcard feature vectors but orthogonal $(k,1)$-wildcard feature vectors.



(a) (b)

Figure 10: **Dependence on parameters for the wildcard kernel.** The graph plots the total number of families for which a given method exceeds an ROC (a) or ROC-50 (b) score threshold. In the graph, the curve of $(5,2,1.0)$-wildcard kernel clearly outperforms the $(5,1,1.0)$-wildcard kernel.

Kernel	Kernel Alignment	ROC	ROC-50
(5,1)-mismatch	0.0982	0.875	0.416
(6,4)-gappy	0.1428	0.851	0.387
(7,5)-gappy	0.0269	0.825	0.315
(8,6)-gappy	0.0090	0.782	0.242
(4,6.0)-substitution	0.1643	0.876	0.441
(5,7.5)-substitution	0.0369	0.865	0.428
(6,9.0)-substitution	0.0170	0.871	0.442
(5,1,1.0)-wildcard	0.0310	0.816	0.304
(5,2,1.0)-wildcard	0.1565	0.881	0.447

Table 1: Mean ROC and ROC-50 scores over 54 target families.

Kernel	$(5,1)$-mismatch	$(6,4)$-gappy	$(4,6)$-subst	$(6,9)$-subst	$(5,1)$-wildcard	$(5,2)$-wildcard
$(5,1)$-mismatch	1.000	0.923	0.812	0.947	0.968	0.864
$(6,4)$-gappy		1.000	0.915	0.742	0.775	0.955
$(4,6)$-subst			1.000	0.591	0.622	0.942
$(6,9)$-subst				1.000	0.991	0.626
$(5,1)$-wildcard					1.000	0.669
$(5,2)$-wildcard						1.000

Table 2: Pairwise kernel alignment scores over the full SCOP data set.

In Table 1, we summarize the mean ROC and ROC-50 scores across the 54 target families for all the string kernels families and parameter values chosen. The table also shows mean training set *kernel-target alignment* scores across the experiments. Kernel alignment was introduced by Cristianini et al. (2001) as a measure of similarity between pairs of kernels or between a kernel and a target function. The *empirical kernel alignment* score between two kernels is defined as the value $\frac{\langle K_1, K_2 \rangle}{\sqrt{\langle K_1, K_1 \rangle \langle K2, K2 \rangle}}$, where K_1 and K_2 are the Gram matrices for the kernels on the sample data, and $\langle \cdot, \cdot \rangle$ is the euclidean inner product when the Gram matrices are viewed as vectors (Hilbert-Schmidt inner product). Thus the alignment score is simply the cosine of the angle between the two vectors representing Gram matrices. The *empirical kernel-target alignment* is the kernel alignment for a Gram matrix and the target $\mathbf{y}\mathbf{y}^t$, where $\mathbf{y}$ is the column vector of labels.

Table 1 shows that for the gappy and wildcard kernels, high kernel-target alignment scores do seem to correlate with good SVM classification performance. However, for the substitution kernels, the kernel-target alignment is low for larger values of k while performance remains strong. In Table 2, we show the pairwise kernel alignment scores between normalized kernels on the full SCOP data set of 7329 sequences. In some cases, the alignment scores between kernels of the same family with different parameters can be quite low, for example the $(5,1,1.0)$-wildcard kernel and $(5,2,1.0)$-wildcard kernel. Surprisingly, the $(6,9)$-substitution kernel Gram matrix is very similar to the $(5,1,1.0)$-wildcard kernel Gram matrix when compared by alignment score, even though their SVM performance is somewhat different, showing that the score gives only a rough measure of kernel similarity. The $(6,4)$-gappy kernel, $(4,6)$-substitution kernel and $(5,2,1.0)$-wildcard kernel are a group of well aligned Gram matrices. $(5,1)$-mismatch kernel seems to be in between the two previous groups in terms of kernel alignment. Clearly, all the models of inexact matching are fairly similar, but there do appear to be several significantly different Gram matrices in the set below that all successfully represent the data for the purposes of SVM learning.

Kernel	ROC	ROC-50
$K_{eq}(\alpha_i = 1)$	0.907	0.520
$K_{opt}(\sigma = 0.01)$	0.901	0.502

Table 3: Mean ROC and ROC-50 scores of linearly combined kernels. Here $K_{opt} = \sum_{i=1}^{N} \alpha_i K_i$, where N is the number of kernels, α is the optimal vector for the best alignment with target yy', and the the regularization parameter depends on σ as described in the text.

Since different kernels capture somewhat different notions of sequence similarity, we consider whether a convex combination of kernels $K(\alpha) = \sum_{i=1}^{N} \alpha_i K_i$, with $\alpha_i \geq 0$ for $i = 1 \ldots N$, can outperform individual kernels. We consider two schemes for choosing such a linear combination. In the first approach, we simply assign equal weights $\alpha_i = 1/N$ for all i to obtain a new kernel K_{eq}. For a second approach, we follow Kandola et al. (2002), who proposed a general method for learning the α_i by solving a optimization problem to maximize the kernel alignment between Gram matrix of $K(\alpha)$ and target yy',

$$A(S, K(\alpha), yy') = \frac{y'K(\alpha)y}{|y||K(\alpha)||},$$

yielding a new kernel K_{opt}. Here, one introduces a regularization parameter λ to constrain $||\alpha||$ and prevent over-alignment; the optimization then amounts to a quadratic programming problem that can be solved through standard methods. We now pick 6 kernels with relatively good performance and low pairwise kernel alignment as components for the new kernel – $(5,1)$-mismatch, $(6,4)$-gappy, $(4,6.0)$-substitution, $(5,7.5)$-substitution, $(6,9.0)$-substitution and $(5,2,1.0)$-wildcard – and repeat the second set of SCOP experiments with these two linear combination kernels. For K_{opt}, we use a regularization parameter of the form $\lambda = \frac{\sigma}{N^2} \sum_{ij} \langle K_i, K_j \rangle$, where $\langle \cdot, \cdot \rangle$ is the Hilbert-Schmidt inner product between matrices. We found that performance varied slightly but significantly as we varied $\sigma = .001, .01, .1, 1, 10, 100, 1000$ (results not shown); since the experiments do not contain a cross-validation set, we simply report the performance of the best parameter choice ($\sigma = .01$) with the caveat that this result may be somewhat optimistic. We report the mean ROC and ROC-50 scores across 54 experiments for the simple case K_{eq}, and the optimal alignment case K_{opt} in Table 3. We found that K_{opt} with the best regularization parameter choice does achieve significant improvement over the best individual kernel (indeed, almost all regularization parameters that we tried displayed some advantage over the best individual kernel); however, the simple weighting used in K_{eq} slightly outperformed K_{opt} in these experiments. Interestingly, for most of 54 experiments, K_{opt} ($\sigma = 0.01$) had non-zero weights only for the two best performing kernels, the $(4,6.0)$-substitution and $(5,2,1.0)$-wildcard kernels, with the weight for the latter about an order of magnitude smaller than that of the former. These results suggest that some of the kernels are complementary to each other and that combining them can help improve performance, though it appears that optimal alignment does not outperform a simple uniform weighting scheme for combining kernels.

6. Discussion

We have presented a number of different k-mer based string kernels that capture a notion of inexact matching – through use of gaps, probabilistic substitutions, and wildcards – but maintain fast computation time. Using a recursive function based on a trie data structure, we show that for all our

new kernels, the time to compute a kernel value $K(x,y)$ is $O(c_K(|x|+|y|))$, where the constant c_K depends on the parameters of the kernel but not on the size of the alphabet Σ. Thus we improve on the constant factor involved in computation of the previously presented mismatch kernel, in which $|\Sigma|$ as well as k and m control the size of the mismatch neighborhood and hence the constant c_K.

We also show how many of our kernels can be obtained through the recently presented transducer formalism of rational $T \circ T^{-1}$ kernels and give the transducer T for several examples. This connection gives an intuitive understanding of the kernel definitions and could inspire new string kernels.

Finally, we present results on two benchmark SCOP data sets for the remote protein homology detection problem and show that many of the new, faster kernels achieve performance comparable to the mismatch kernel. We also investigate how kernel performance depends on parameter choice for the different inexact matching models. Intuitively, it is clear that the only biological reasonable choices involve short k-mer features, since as we allow k to grow, we cannot permit sufficient inexact matching without also introducing noise. However, within these constraints, our results demonstrate the somewhat different behavior of the various kernel families.

We note that Vishwanathan and Smola (2002) used counting statistics and a suffix tree construction to eliminate the constant factor of k in computation time for the exact-matching spectrum kernel (Leslie et al., 2002a). It may be possible to extend this technique to the fast inexact-matching kernels presented here.

A promising direction for applied work in this area is combining string kernel representations with semi-supervised approaches for leveraging the abundant unlabeled protein sequence data (sequences whose 3D structure is unknown) available in sequence databases. One recent approach is presented by Weston et al. (2003), where string kernels are used as a base kernel representation, and unlabeled sequence data together with a dissimilarity measure between sequence examples are used to build *cluster kernels* that modify the base kernel for a richer representation. In more recent work (Kuang et al., 2004), we define k-mer based string kernels for probabilistic sequence profiles (Gribskov et al., 1987), which also give a richer representation of sequences by estimating position specific residue emission probabilities from unlabeled data. These profile-based string kernels provide another promising semi-supervised approach for kernel representation of protein sequence data.

Acknowledgments

We would like to thank Eleazar Eskin, Risi Kondor and William Stafford Noble for helpful discussions and Corinna Cortes, Patrick Haffner and Mehryar Mohri for explaining their transducer formalism to us. This work is supported by an Award in Informatics from the PhRMA Foundation, NIH grant LM07276-02, and NSF grant ITR-0312706.

References

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

C. Cortes, P. Haffner, and M. Mohri. Rational kernels. *Neural Information Processing Systems 16*, 2002.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge, 2000.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *Neural Information Processing Systems*, volume 15, 2001.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge UP, 1998.

E. Eskin, W. S. Noble, Y. Singer, and S. Snir. A unified approach for sequence prediction using sparse sequence models. Technical report, Hebrew University, 2003.

M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: Detection of distantly related proteins. *PNAS*, pages 4355–4358, 1987.

M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.

D. Haussler. Convolution kernels on discrete structure. Technical report, UC Santa Cruz, 1999.

S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *PNAS*, 89: 10915–10919, 1992.

T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 149–158. AAAI Press, 1999.

J. Kandola, J. Shawe-Taylor, and N. Cristianini. Optimizing kernel alignment over combinations of kernels. NeoroCOLTTechnicalReport NC-TR-2002-121, http://www.neurocolt.org, 2002.

R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. In *Computational Systems Bioinformatics*, 2004.

C. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 2004.

C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. *Proceedings of the Pacific Biocomputing Symposium*, 2002a.

C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. *Neural Information Processing Systems 16*, 2002b.

C. Leslie and R. Kuang. Fast kernels for inexact string matching. *Proceedings of COLT/Kernel Workshop*, 2003.

C. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology*, 2002.

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.

A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

M. Sagot. Spelling approximate or repeated motifs using a suffix tree. *Lecture Notes in Computer Science*, 1380:111–127, 1998.

S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:371–328, 1997.

R. M. Schwartz and M. O. Dayhoff. Matrices for detecing distant relationships. In *Atlas of Protein Sequence and Structure*, pages 353–358, Silver Spring, MD, 1978. National Biomedical Research Foundation.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

J.-P. Vert, H. Saigo, and T. Akutsu. *Kernel Methods in Computational Biology*, chapter Local alignment kernels for biological sequences. MIT Press, 2004.

S. V. N. Vishwanathan and A. Smola. Fast kernels for string and tree matching. *Neural Information Processing Systems 16*, 2002.

M. S. Waterman, J. Joyce, and M. Eggert. *Computer alignment of sequences*, chapter Phylogenetic Analysis of DNA Sequences. Oxford, 1991.

C. Watkins. Dynamic alignment kernels. Technical report, UL Royal Holloway, 1999.

J. Weston, C. Leslie, D. Zhou, A. Elisseeff, and W. S. Noble. Cluster kernels for semi-supervised protein classification. *Neural Information Processing Systems 17*, 2003.

Journal of Machine Learning Research 5 (2004) 1457–1469 Submitted 8/04; Published 11/04

Non-negative Matrix Factorization with Sparseness Constraints

Patrik O. Hoyer PATRIK.HOYER@HELSINKI.FI
HIIT Basic Research Unit
Department of Computer Science
P.O. Box 68, FIN-00014
University of Helsinki
Finland

Editor: Peter Dayan

Abstract

Non-negative matrix factorization (NMF) is a recently developed technique for finding parts-based, linear representations of non-negative data. Although it has successfully been applied in several applications, it does not always result in parts-based representations. In this paper, we show how explicitly incorporating the notion of 'sparseness' improves the found decompositions. Additionally, we provide complete MATLAB code both for standard NMF and for our extension. Our hope is that this will further the application of these methods to solving novel data-analysis problems.

Keywords: non-negative matrix factorization, sparseness, data-adaptive representations

1. Introduction

A fundamental problem in many data-analysis tasks is to find a suitable representation of the data. A useful representation typically makes latent structure in the data explicit, and often reduces the dimensionality of the data so that further computational methods can be applied.

Non-negative matrix factorization (NMF) (Paatero and Tapper, 1994; Lee and Seung, 1999) is a recent method for finding such a representation. Given a non-negative data matrix $\mathbf{V}$, NMF finds an approximate factorization $\mathbf{V} \approx \mathbf{WH}$ into non-negative factors $\mathbf{W}$ and $\mathbf{H}$. The non-negativity constraints make the representation purely additive (allowing no subtractions), in contrast to many other linear representations such as principal component analysis (PCA) and independent component analysis (ICA) (Hyvärinen et al., 2001).

One of the most useful properties of NMF is that it usually produces a *sparse* representation of the data. Such a representation encodes much of the data using few 'active' components, which makes the encoding easy to interpret. Sparse coding (Field, 1994) has also, on theoretical grounds, been shown to be a useful middle ground between completely distributed representations, on the one hand, and unary representations (grandmother cells) on the other (Földiák and Young, 1995; Thorpe, 1995). However, because the sparseness given by NMF is somewhat of a side-effect rather than a goal, one cannot in any way control the degree to which the representation is sparse. In many applications, more direct control over the properties of the representation is needed.

In this paper, we extend NMF to include the option to control sparseness explicitly. We show that this allows us to discover parts-based representations that are qualitatively better than those given

by basic NMF. We also discuss the relationship between our method and other recent extensions of NMF (Li et al., 2001; Hoyer, 2002; Liu et al., 2003).

Additionally, this contribution includes a complete MATLAB package for performing NMF and its various extensions. Although the most basic version of NMF requires only two lines of code and certainly does not warrant distributing a separate software package, its several extensions involve more complicated operations; the absense of ready-made code has probably hindered their widespread use so far. We hope that our software package will alleviate the problem.

This paper is structured as follows. In Section 2 we describe non-negative matrix factorization, and discuss its success but also its limitations. Section 3 discusses why and how to incorporate sparseness constraints into the NMF formulation. Section 4 provides experimental results that verify our approach. Finally, Sections 5 and 6 compare our approach to other recent extensions of NMF and conclude the paper.

2. Non-negative Matrix Factorization

Non-negative matrix factorization is a *linear*, *non-negative* approximate data representation. Let's assume that our data consists of T measurements of N non-negative scalar variables. Denoting the (N-dimensional) measurement vectors $\mathbf{v}^t$ ($t = 1, \ldots, T$), a linear approximation of the data is given by

$$\mathbf{v}^t \approx \sum_{i=1}^{M} \mathbf{w}_i h_i^t = \mathbf{W}\mathbf{h}^t,$$

where $\mathbf{W}$ is an $N \times M$ matrix containing the *basis vectors* $\mathbf{w}_i$ as its columns. Note that each measurement vector is written in terms of the *same* basis vectors. The M basis vectors $\mathbf{w}_i$ can be thought of as the 'building blocks' of the data, and the (M-dimensional) coefficient vector $\mathbf{h}^t$ describes how strongly each building block is present in the measurement vector $\mathbf{v}^t$.

Arranging the measurement vectors $\mathbf{v}^t$ into the columns of an $N \times T$ matrix $\mathbf{V}$, we can now write

$$\mathbf{V} \approx \mathbf{W}\mathbf{H},$$

where each column of $\mathbf{H}$ contains the coefficient vector $\mathbf{h}^t$ corresponding to the measurement vector $\mathbf{v}^t$. Written in this form, it becomes apparent that a linear data representation is simpy a factorization of the data matrix. Principal component analysis, independent component analysis, vector quantization, and non-negative matrix factorization can all be seen as matrix factorization, with different choices of objective function and/or constraints.

Whereas PCA and ICA do not in any way restrict the signs of the entries of $\mathbf{W}$ and $\mathbf{H}$, NMF requires all entries of both matrices to be non-negative. What this means is that the data is described by using additive components only. This constraint has been motivated in a couple of ways. First, in many applications one knows (for example by the rules of physics) that the quantities involved cannot be negative. In such cases, it can be difficult to interpret the results of PCA and ICA (Paatero and Tapper, 1994; Parra et al., 2000). Second, non-negativity has been argued for based on the intuition that parts are generally combined additively (and not subtracted) to form a whole; hence, these constraints might be useful for learning parts-based representations (Lee and Seung, 1999).

Given a data matrix $\mathbf{V}$, the optimal choice of matrices $\mathbf{W}$ and $\mathbf{H}$ are defined to be those non-negative matrices that minimize the reconstruction error between $\mathbf{V}$ and $\mathbf{WH}$. Various error functions have been proposed (Paatero and Tapper, 1994; Lee and Seung, 2001), perhaps the most widely

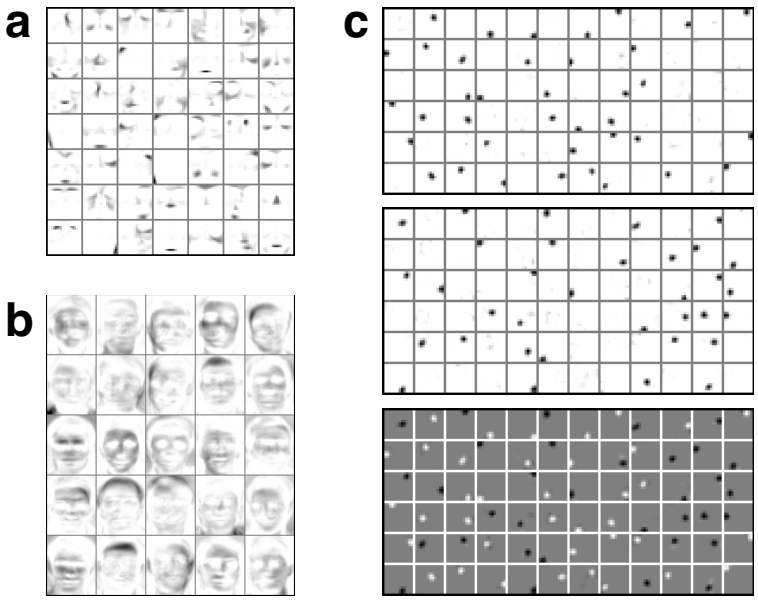


Figure 1: NMF applied to various image data sets. **(a)** Basis images given by NMF applied to face image data from the CBCL database (`http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html`), following Lee and Seung (1999). In this case NMF produces a parts-based representation of the data. **(b)** Basis images derived from the ORL face image database (`http://www.uk.research.att.com/facedatabase.html`), following Li et al. (2001). Here, the NMF representation is global rather than parts-based. **(c)** Basis vectors from NMF applied to ON/OFF-contrast filtered natural image data (Hoyer, 2003). Top: Weights for the ON-channel. Each patch represents the part of one basis vector $\mathbf{w}_i$ corresponding to the ON-channel. (White pixels denote zero weight, darker pixels are positive weights.) Middle: Corresponding weights for the OFF-channel. Bottom: Weights for ON minus weights for OFF. (Here, gray pixels denote zero.) Note that NMF represents this natural image data using circularly symmetric features.

used is the squared error (euclidean distance) function

$$E(\mathbf{W},\mathbf{H}) = \|\mathbf{V}-\mathbf{WH}\|^2 = \sum_{i,j}(V_{ij}-(\mathbf{WH})_{ij})^2.$$

Although the minimization problem is convex in $\mathbf{W}$ and $\mathbf{H}$ separately, it is not convex in both simultaneously. Paatero and Tapper (1994) gave a gradient algorithm for this optimization, whereas Lee and Seung (2001) devised a multiplicative algorithm that is somewhat simpler to implement and also showed good performance.

Although some theoretical work on the properties of the NMF representation exists (Donoho and Stodden, 2004), much of the appeal of NMF comes from its empirical success in learning meaningful features from a diverse collection of real-life data sets. Lee and Seung (1999) showed that, when the data set consisted of a collection of face images, the representation consisted of

basis vectors encoding for the mouth, nose, eyes, etc; the intuitive features of face images. In Figure 1a we have reproduced that basic result using the same data set. Additionally, they showed that meaningful topics can be learned when text documents are used as data. Subsequently, NMF has been successfully applied to a variety of data sets (Buchsbaum and Bloch, 2002; Brunet et al., 2004; Jung and Kim, 2004; Kim and Tidor, 2003).

Despite this success, there also exist data sets for which NMF does not give an intuitive decomposition into parts that would correspond to our idea of the 'building blocks' of the data. Li et al. (2001) showed that when NMF was applied to a different facial image database, the representation was global rather than local, qualitatively different from that reported by Lee and Seung (1999). Again, we have rerun that experiment and confirm those results, see Figure 1b. The difference was mainly attributed to how well the images were hand-aligned (Li et al., 2001).

Another case where the decomposition found by NMF does not match the underlying elements of the data is shown in Figure 1c. In this experiment (Hoyer, 2003), natural image patches were high-pass filtered and subsequently split into positive ('ON') and negative ('OFF') contrast channels, in a process similar to how visual information is processed by the retina. When NMF is applied to such a data set, the resulting decomposition does not consist of the oriented filters which form the cornerstone of most of modern image processing. Rather, NMF represents these images using simple, dull, circular 'blobs'.

We will show that, in both of the above cases, explicitly controlling the sparseness of the representation leads to representations that are parts-based and match the intuitive features of the data.

3. Adding Sparseness Constraints to NMF

In this section, we describe the basic idea of sparseness, and show how to incorporate it into the NMF framework.

3.1 Sparseness

The concept of 'sparse coding' refers to a representational scheme where only a few units (out of a large population) are effectively used to represent typical data vectors (Field, 1994). In effect, this implies most units taking values close to zero while only few take significantly non-zero values. Figure 2 illustrates the concept and our sparseness measure (defined below).

Numerous sparseness measures have been proposed and used in the literature to date. Such measures are mappings from $\mathbb{R}^n$ to $\mathbb{R}$ which quantify how much energy of a vector is packed into only a few components. On a normalized scale, the sparsest possible vector (only a single component is non-zero) should have a sparseness of one, whereas a vector with all elements equal should have a sparseness of zero.

In this paper, we use a sparseness measure based on the relationship between the L_1 norm and the L_2 norm:

$$\text{sparseness}(\mathbf{x}) = \frac{\sqrt{n} - (\sum |x_i|) / \sqrt{\sum x_i^2}}{\sqrt{n} - 1},$$

where n is the dimensionality of $\mathbf{x}$. This function evaluates to unity if and only if $\mathbf{x}$ contains only a single non-zero component, and takes a value of zero if and only if all components are equal (up to signs), interpolating smoothly between the two extremes.

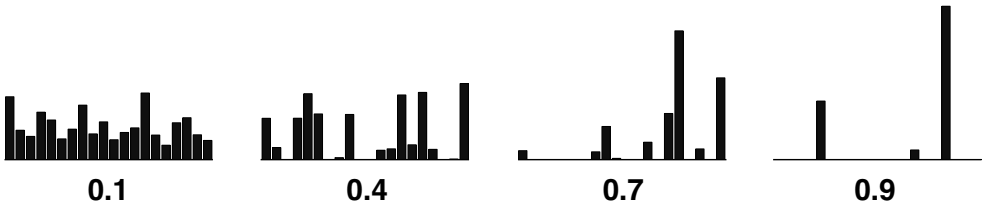


Figure 2: Illustration of various degrees of sparseness. Four vectors are shown, exhibiting sparseness levels of 0.1, 0.4, 0.7, and 0.9. Each bar denotes the value of one element of the vector. At low levels of sparseness (leftmost), all elements are roughly equally active. At high levels (rightmost), most coefficients are zero whereas only a few take significant values.

3.2 NMF with Sparseness Constraints

Our aim is to constrain NMF to find solutions with desired degrees of sparseness. The first question to answer is then: what exactly should be sparse? The basis vectors $\mathbf{W}$ or the coefficients $\mathbf{H}$? This is a question that cannot be given a general answer; it all depends on the specific application in question. Further, just transposing the data matrix switches the role of the two, so it is easy to see that the choice of which to constrain (or both, or none) must be made by the experimenter.

For example, a doctor analyzing disease patterns might assume that most diseases are rare (hence sparse) but that each disease can cause a large number of symptoms. Assuming that symptoms make up the rows of her matrix and the columns denote different individuals, in this case it is the 'coefficients' which should be sparse and the 'basis vectors' unconstrained. On the other hand, when trying to learn useful features from a database of images, it might make sense to require both $\mathbf{W}$ and $\mathbf{H}$ to be sparse, signifying that any given object is *present* in few images and *affects* only a small part of the image.

These considerations lead us to defining NMF with sparseness constraints as follows:

Definition: NMF with sparseness constraints

Given a non-negative data matrix $\mathbf{V}$ of size $N \times T$, find the non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ of sizes $N \times M$ and $M \times T$ (respectively) such that

$$E(\mathbf{W},\mathbf{H}) = \|\mathbf{V} - \mathbf{W}\mathbf{H}\|^2 \tag{1}$$

is minimized, under *optional constraints*

$$\begin{aligned} \text{sparseness}(\mathbf{w}_i) &= S_w, \ \forall i \\ \text{sparseness}(\mathbf{h}_i) &= S_h, \ \forall i, \end{aligned}$$

where $\mathbf{w}_i$ is the i:th *column* of $\mathbf{W}$ and $\mathbf{h}_i$ is the i:th *row* of $\mathbf{H}$. Here, M denotes the number of components, and S_w and S_h are the desired sparsenesses of $\mathbf{W}$ and $\mathbf{H}$ (respectively). These three parameters are set by the user.

Note that we did not constrain the scales of $\mathbf{w}_i$ or $\mathbf{h}_i$ yet. However, since $\mathbf{w}_i\mathbf{h}_i = (\mathbf{w}_i\lambda)(\mathbf{h}_i/\lambda)$ we are free to arbitrarily fix any norm of either one. In our algorithm, we thus choose to fix the L_2 norm of $\mathbf{h}_i$ to unity, as a matter of convenience.

3.3 Algorithm

We have devised a projected gradient descent algorithm for NMF with sparseness constraints. This algorithm essentially takes a step in the direction of the negative gradient, and subsequently projects onto the constraint space, making sure that the taken step is small enough that the objective function (1) is reduced at every step. The main muscle of the algorithm is the projection operator which enforces the desired degree of sparseness. This operator is described in detail following this algorithm.

Algorithm: NMF with sparseness constraints

1. Initialize $\mathbf{W}$ and $\mathbf{H}$ to random positive matrices
2. If sparseness constraints on $\mathbf{W}$ apply, then project each column of $\mathbf{W}$ to be non-negative, have unchanged L_2 norm, but L_1 norm set to achieve desired sparseness
3. If sparseness constraints on $\mathbf{H}$ apply, then project each row of $\mathbf{H}$ to be non-negative, have unit L_2 norm, and L_1 norm set to achieve desired sparseness
4. Iterate
   (a) If sparseness constraints on $\mathbf{W}$ apply,
      i. Set $\mathbf{W} := \mathbf{W} - \mu_{\mathbf{W}}(\mathbf{WH} - \mathbf{V})\mathbf{H}^T$
      ii. Project each column of $\mathbf{W}$ to be non-negative, have unchanged L_2 norm, but L_1 norm set to achieve desired sparseness

   else take standard multiplicative step $\mathbf{W} := \mathbf{W} \otimes (\mathbf{VH}^T) \oslash (\mathbf{WHH}^T)$

   (b) If sparseness constraints on $\mathbf{H}$ apply,
      i. Set $\mathbf{H} := \mathbf{H} - \mu_{\mathbf{H}}\mathbf{W}^T(\mathbf{WH} - \mathbf{V})$
      ii. Project each row of $\mathbf{H}$ to be non-negative, have unit L_2 norm, and L_1 norm set to achieve desired sparseness

   else take standard multiplicative step $\mathbf{H} := \mathbf{H} \otimes (\mathbf{W}^T\mathbf{V}) \oslash (\mathbf{W}^T\mathbf{WH})$

Above, $\otimes$ and $\oslash$ denote elementwise multiplication and division, respectively. Moreover, $\mu_{\mathbf{W}}$ and $\mu_{\mathbf{H}}$ are small positive constants (stepsizes) which must be set appropriately for the algorithm to work. Fortunately, they need not be set by the user; our implementation of the algorithm automatically adapts these parameters. The multiplicative steps are directly taken from Lee and Seung (2001) and are used when constraints are not to be applied.

Many of the steps in the above algorithm require a projection operator which enforces sparseness by explicitly setting both L_1 and L_2 norms (and enforcing non-negativity). This operator is defined as follows

---

problem Given any vector $\mathbf{x}$, find the closest (in the euclidean sense) *non-negative* vector $\mathbf{s}$ with a given L_1 norm and a given L_2 norm.

algorithm The following algorithm solves the above problem. See below for comments.

1. Set $s_i := x_i + (L_1 - \sum x_i)/\mathrm{dim}(\mathbf{x}),\ \forall i$
2. Set $Z := \{\}$
3. Iterate
   (a) Set $m_i := \begin{cases} L_1/(\mathrm{dim}(\mathbf{x}) - \mathrm{size}(Z)) & \text{if } i \notin Z \\ 0 & \text{if } i \in Z \end{cases}$
   (b) Set $\mathbf{s} := \mathbf{m} + \alpha(\mathbf{s} - \mathbf{m})$, where $\alpha \geq 0$ is selected such that the resulting $\mathbf{s}$ satisfies the L_2 norm constraint. This requires solving a quadratic equation.
   (c) If all components of $\mathbf{s}$ are non-negative, return $\mathbf{s}$, end
   (d) Set $Z := Z \cup \{i; s_i < 0\}$
   (e) Set $s_i := 0,\ \forall i \in Z$
   (f) Calculate $c := (\sum s_i - L_1)/(\mathrm{dim}(\mathbf{x}) - \mathrm{size}(Z))$
   (g) Set $s_i := s_i - c,\ \forall i \notin Z$
   (h) Go to (a)

In words, the above algorithm works as follows: We start by projecting the given vector onto the hyperplane $\sum s_i = L_1$. Next, within this space, we project to the closest point on the joint constraint hypersphere (intersection of the sum and the L_2 constraints). This is done by moving radially outward from the center of the sphere (the center is given by the point where all components have equal values). If the result is completely non-negative, we have arrived at our destination. If not, those components that attained negative values must be fixed at zero, and a new point found in a similar fashion under those additional constraints.

Note that, once we have a solution to the above *non-negative* problem, it would be straightforward to extend it to a general solution without non-negativity constraints. If a given component of $\mathbf{x}$ is positive (negative), we know because of the symmetries of L_1 and L_2 norms that the optimal solution $\mathbf{s}$ will have the corresponding component positive or zero (negative or zero). Thus, we may simply record the signs of $\mathbf{x}$, take the absolute value, perform the projection in the first quadrant using the algorithm above, and re-enter the signs into the solution.

In principle, the devised projection algorithm may take as many as $\mathrm{dim}(\mathbf{x})$ iterations to converge to the correct solution (because at each iteration the algorithm either converges, or at least one component is added to the set of zero valued components). In practice, however, the algorithm converges much faster. In Section 4 we show that even for extremely high dimensions the algorithm typically converges in only a few iterations.

3.4 Matlab Implementation

Our software package, available at `http://www.cs.helsinki.fi/patrik.hoyer/` implements all the details of the above algorithm. In particular, we monitor the objective function E throughout the optimization, and adapt the stepsizes to ensure convergence. The software package contains, in

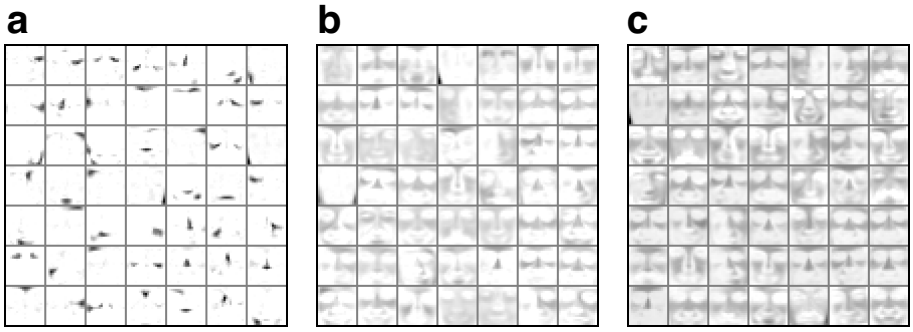


Figure 3: Features learned from the CBCL face image database using NMF with sparseness constraints. **(a)** The sparseness of the basis images were fixed to 0.8, slightly higher than the average sparseness produced by standard NMF, yielding a similar result. The sparseness of the coefficients was unconstrained. **(b)** Here, we switched the sparseness constraints such that the coefficients were constrained to 0.8 but the basis images were unconstrained. Note that this creates a global representation similar to that given by vector quantization (Lee and Seung, 1999). **(c)** Illustration of another way to obtain a global representation: setting the sparseness of the basis images to a low value (here: 0.2) also yields a non-local representation.

addition to the projection operator and NMF code, all the files needed to reproduce the results described in this paper, with the exception of data sets. For copyright reasons the face image databases are not included, but they can easily be downloaded separately from their respective www addresses.

4. Experiments with Sparseness Constraints

In this section, we show that adding sparseness constraints to NMF can make it find parts-based representations in cases where unconstrained NMF does not. In addition, we experimentally verify our claim that the projection operator described in Section 3.3 converges in only a few iterations even when the dimensionality of the vector is high.

4.1 Representations Learned from Face Image Databases

Recall from Section 2 the mixed results of applying standard NMF to face image data. Lee and Seung (1999) originally showed that NMF found a parts-based representation when trained on data from the CBCL database. However, when applied to the ORL data set, in which images are not as well aligned, a global decomposition emerges. These results were shown in Figure 1a and 1b. To compare, we applied sparseness constrained NMF to both face image data sets.

For the CBCL data, some resulting bases are shown in Figure 3. Setting a high sparseness value for the basis images results in a local representation similar to that found by standard NMF. However, we want to emphasize the fact that sparseness constrained NMF does not always lead to local solutions: Global solutions can be obtained by deliberately setting a low sparseness on the basis images, or by requiring a high sparseness on the coefficients (forcing each coefficient to try to represent more of the image).

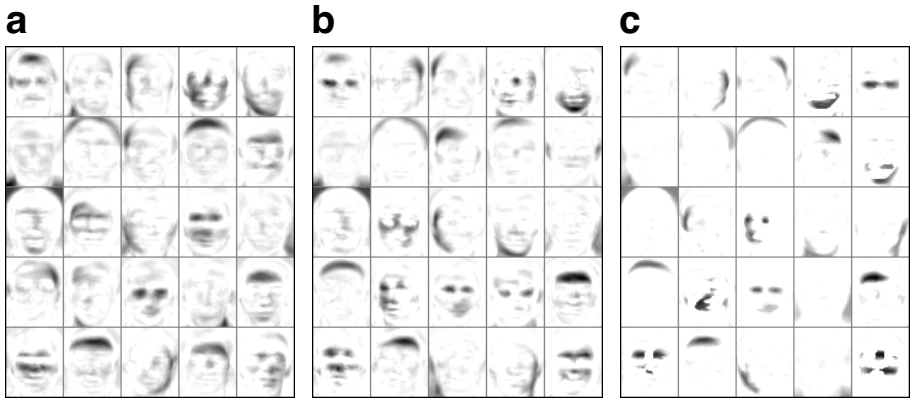


Figure 4: Features learned from the ORL face image database using NMF with sparseness constraints. When increasing the sparseness of the basis images, the representation switches from a global one (like the one given by standard NMF, cf Figure 1b) to a local one. Sparseness levels were set to **(a)** 0.5 **(b)** 0.6 **(c)** 0.75.

The ORL database provides the more interesting test of the method. In Figure 4 we show bases learned by sparseness constrained NMF, for various sparseness settings. Note that our method can learn a parts-based representation of this data set, in contrast to standard NMF. Also note that the representation is not very sensitive to the specific sparseness level chosen.

4.2 Basis Derived from Natural Image Patches

In Figure 1c we showed that standard NMF applied to natural image data produces only circular features, not oriented features like those employed by modern image processing techniques. Here, we tested the result of using additional sparseness constraints. Figure 5 shows the basis vectors obtained by putting a sparseness constraint on the coefficients ($S_h = 0.85$) but leaving the sparseness of the basis vectors unconstrained. In this case, NMF learns oriented features that represent edges and lines. Such oriented features are widely regarded as the best type of low-level features for representing natural images, and similar features are also used by the early visual system of the biological brain (Field, 1987; Simoncelli et al., 1992; Olshausen and Field, 1996; Bell and Sejnowski, 1997). This example illustrates that sparseness constrained NMF does not simply 'sparsify' the result of standard, unconstrained NMF, but rather can find qualitatively different parts-based representations that are more compatible with the sparseness assumptions.

4.3 Convergence of Algorithm Implementing the Projection Step

To verify the performance of our projection method we performed extensive tests, varying the number of dimensions, the desired degree of sparseness, and the sparseness of the original vector. The desired and the initial degrees of sparseness were set to 0.1, 0.3, 0.5, 0.7, and 0.9, and the dimensionality of the problem was set to 2, 3, 5, 10, 50, 100, 500, 1000, 3000, 5000, and 10000. All combinations of sparsenesses and dimensionalities were analyzed. Based on this analysis, the worst case (most iterations on average required) was when the desired degree of sparseness was high (0.9)

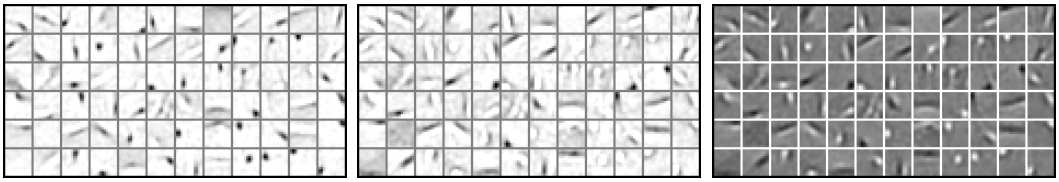


Figure 5: Basis vectors from ON/OFF-filtered natural images obtained using NMF with sparseness constraints. The sparseness of the coefficients was fixed at 0.85, and the sparseness of the basis images was unconstrained. As opposed to standard NMF (cf Figure 1c), the representation is based on oriented, Gabor-like, features.

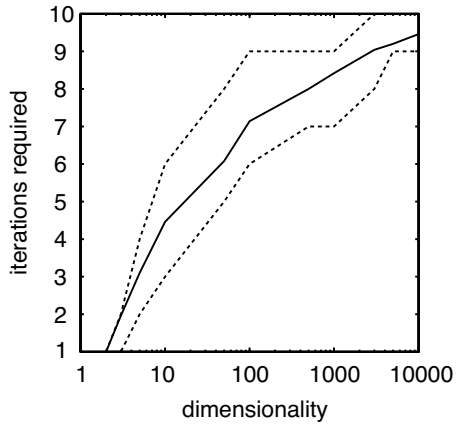


Figure 6: Number of iterations required for the projection algorithm to converge, in the worst-case scenario tested (desired sparseness 0.9, initial sparseness 0.1). The solid line shows the average number (over identical random trials) of iterations required, the dashed lines show the minimum and maximum iterations. Note that the number of iterations grows very slowly with the dimensionality of the problem.

but the initial sparseness was low (0.1). In Figure 6 we plots the number of iterations required for this worst case, as a function of dimensionality. Even in this worst-case scenario, and even for the highest tested dimensionality, the algorithm never required more than 10 iterations to converge. Thus, although we do not have analytical bounds on the performance on the algorithm, empirically the projection method performs extremely well.

5. Relation to Other Recent Work

Here, we describe how our method relates to other recently developed extensions of NMF and to non-negative independent component analysis.

5.1 Extensions of NMF

Several authors have noted the shortcomings of standard NMF, and suggested extensions and modifications of the original model. Li et al. (2001) noted that NMF found only global features from the ORL database (see Figure 1b) and suggested an extension they call *Local* Non-negative Matrix Factorization (LNMF). Their method indeed produces local features from the ORL database, similar to those given by our method (Figure 4c). However, it does not produce oriented filters from natural image data (results not shown). Further, there is no way to explicitly control the sparseness of the representation, should this be needed.

Hoyer (2002) extended the NMF framework to include an adjustable sparseness parameter. The present paper is an extension of those ideas. The main improvement is that in the present model sparseness is adjusted explicitly, rather than implicitly. This means that one does not any more need to employ trial-and-error to find the parameter setting that yields the desired level of sparseness.

Finally, Liu et al. (2003) also noted the need for incorporating the notion of sparseness, and suggested an extension termed *Sparse* Non-negative Matrix Factorization (SNMF). Their extension is similar in spirit and form to that given by Hoyer (2002) with the added benefit of yielding a more convenient, faster algorithm. Nevertheless, it also suffers from the drawback that sparseness is only controlled implicitly. Furthermore, their method does not yield oriented features from natural image data (results not shown).

In summary, the framework presented in the present paper improves on these previous extensions by allowing explicit control of the statistical properties of the representation.

In order to facilitate the use of, and comparison between, the various extensions of NMF, they are all provided as part of the Matlab code package distributed with this paper. Using this package readers can effortlessly verify our current claims by applying the algorithms to the various data sets. Moreover, the methods can be compared head-to-head on new interesting data sets.

5.2 Non-negative Independent Component Analysis

Our method has a close connection to the statistical technique called independent component analysis (ICA) (Hyvärinen et al., 2001). ICA attempts to find a matrix factorization similar to ours, but with two important differences. First, the signs of the components are in general not restricted; in fact, symmetry is often assumed, implying an approximately equal number of positive and negative elements. Second, the sources are not forced to any desired degree of sparseness (as in our method) but rather sparseness is incorporated into the objective function to be optimized. The sparseness goal can be put on either $\mathbf{W}$ or $\mathbf{H}$, or both (Stone et al., 2002).

Recently, some authors have considered estimating the ICA model in the case of one-sided, non-negative sources (Plumbley, 2003; Oja and Plumbley, 2004). In these methods, non-negativity is not specified as a constraint but rather as an objective; hence, complete non-negativity of the representation is seldom achieved for real-life data sets. Nevertheless, one can show that if the linear ICA model holds, with non-negative components, these methods can identify the model.

6. Conclusions

Non-negative matrix factorization (NMF) has proven itself a useful tool in the analysis of a diverse range of data. One of its most useful properties is that the resulting decompositions are often intuitive and easy to interpret because they are sparse. Sometimes, however, the sparseness achieved

by NMF is not enough; in such situations it might be useful to control the degree of sparseness explicitly. Our main contributions of this paper were (a) to describe a projection operator capable of simultaneously enforcing both L_1 and L_2 norms and hence any desired degree of sparseness, (b) to show its use in the NMF framework for learning representations that could not be obtained by regular NMF, and (c) to provide a software package to enable researchers and practitioners to easily perform NMF and its various extensions. We hope that all three contributions will prove useful to the field of data-analysis.

Acknowledgments

The author wishes to thank Jarmo Hurri, Aapo Hyvärinen, and Fabian Theis for useful discussions and comments on the manuscript.

References

A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.

J. P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences*, 101:4164–4169, 2004.

G. Buchsbaum and O. Bloch. Color categories revealed by non-negative matrix factorization of munsell color spectra. *Vision Research*, 42:559–563, 2002.

D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing 16 (Proc. NIPS*2003)*. MIT Press, 2004.

D. J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America*, 4:2379–2394, 1987.

D. J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

P. Földiák and M. P. Young. Sparse coding in the primate cortex. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 895–898. The MIT Press, Cambridge, Massachusetts, 1995.

P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing XII (Proc. IEEE Workshop on Neural Networks for Signal Processing)*, pages 557–565, Martigny, Switzerland, 2002.

P. O. Hoyer. Modeling receptive fields with non-negative sparse coding. In E. De Schutter, editor, *Computational Neuroscience: Trends in Research 2003*. Elsevier, Amsterdam, 2003. Also published in: Neurocomputing 52-54 (2003), pp 547-552.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley Interscience, 2001.

K. Jung and E. Y. Kim. Automatic text extraction for content-based image indexing. *Advances in Knowledge Discovery and Data Mining: Proceedings Lecture Notes in Artificial Intelligence*, 3056:497–507, 2004.

P. M. Kim and B. Tidor. Subsystem identification through dimensionality reduction of large-scale gene expression data. *Genome Research*, 13:1706–1718, 2003.

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. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing 13 (Proc. NIPS*2000)*. MIT Press, 2001.

S. Z. Li, X. Hou, H. Zhang, and Q. Cheng. Learning spatially localized parts-based representations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Vol. I*, pages 207–212, Hawaii, USA, 2001.

W. Liu, N. Zheng, and X. Lu. Non-negative matrix factorization for visual coding. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'2003)*, 2003.

E. Oja and M. Plumbley. Blind separation of positive sources by globally convergent gradient search. *Neural Computation*, 16(9):1811–1825, 2004.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

L. Parra, C. Spence, P. Sajda, A. Ziehe, and K.-R. Müller. Unmixing hyperspectral data. In *Advances in Neural Information Processing 12 (Proc. NIPS*99)*, pages 942–948. MIT Press, 2000.

M. Plumbley. Algorithms for non-negative independent component analysis. *IEEE Transactions on Neural Networks*, 14(3):534–543, 2003.

E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multiscale transforms. *IEEE Transactions on Information Theory*, 38:587–607, 1992.

J. V. Stone, J. Porrill, N. R. Porter, and I. D. Wilkinson. Spatiotemporal independent component analysis of event-related fMRI data using skewed probability density functions. *Neuroimage*, 15: 407–421, 2002.

S. Thorpe. Localized versus distributed representations. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 549–552. The MIT Press, Cambridge, Massachusetts, 1995.

Journal of Machine Learning Research 5 (2004) 1471–1530 Submitted 10/03; Revised 10/04; Published 11/04

Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning

Evan Greensmith EVAN@CSL.ANU.EDU.AU
Research School of Information Sciences and Engineering
Australian National University
Canberra 0200, Australia

Peter L. Bartlett BARTLETT@STAT.BERKELEY.EDU
Computer Science Division & Department of Statistics
UC Berkeley
Berkeley, CA 94720, USA

Jonathan Baxter JBAXTER@PANSCIENT.COM
Panscient Pty. Ltd.
10 Gawler Terrace
Walkerville, SA 5081, Australia

Editor: Michael Littman

Abstract

Policy gradient methods for reinforcement learning avoid some of the undesirable properties of the value function approaches, such as policy degradation (Baxter and Bartlett, 2001). However, the variance of the performance gradient estimates obtained from the simulation is sometimes excessive. In this paper, we consider variance reduction methods that were developed for Monte Carlo estimates of integrals. We study two commonly used policy gradient techniques, the baseline and actor-critic methods, from this perspective. Both can be interpreted as additive control variate variance reduction methods. We consider the expected average reward performance measure, and we focus on the GPOMDP algorithm for estimating performance gradients in partially observable Markov decision processes controlled by stochastic reactive policies. We give bounds for the estimation error of the gradient estimates for both baseline and actor-critic algorithms, in terms of the sample size and mixing properties of the controlled system. For the baseline technique, we compute the optimal baseline, and show that the popular approach of using the average reward to define the baseline can be suboptimal. For actor-critic algorithms, we show that using the true value function as the critic can be suboptimal. We also discuss algorithms for estimating the optimal baseline and approximate value function.

Keywords: reinforcement learning, policy gradient, baseline, actor-critic, GPOMDP

1. Introduction

The task in reinforcement learning problems is to select a controller that will perform well in some given environment. This environment is often modelled as a partially observable Markov decision process (POMDP); see, for example, Kaelbling et al. (1998); Aberdeen (2002); Lovejoy (1991). At any step in time this process sits in some state, and that state is updated when the POMDP is supplied with an action. An observation is generated from the current state and given as information to a controller. A reward is also generated, as an indication of how good that state is to be in.

The controller can use the observations to determine which action to produce, thereby altering the POMDP state. The expectation of the average reward over possible future sequences of states given a particular controller (the expected average reward) can be used as a measure of how well a controller performs. This performance measure can then be used to select a controller that will perform well.

Given a parameterized space of controllers, one method to select a controller is by gradient ascent (see, for example, Glynn, 1990; Glynn and L'Ecuyer, 1995; Reiman and Weiss, 1989; Rubinstein, 1991; Williams, 1992). An initial controller is selected, then the gradient direction in the controller space of the expected average reward is calculated. The gradient information can then be used to find the locally optimal controller for the problem. The benefit of using a gradient approach, as opposed to directly comparing the expected average reward at different points, is that it can be less susceptible to error in the presence of noise. The noise arises from the fact that we estimate, rather than calculate, properties of the controlled POMDP.

Determining the gradient requires the calculation of an integral. We can produce an estimate of this integral through Monte Carlo techniques. This changes the integration problem into one of calculating a weighted average of samples. It turns out that these samples can be generated purely by watching the controller act in the environment (see Section 3.3). However, this estimation tends to have a high variance associated with it, which means a large number of steps is needed to obtain a good estimate.

GPOMDP (Baxter and Bartlett, 2001) is an algorithm for generating an estimate of the gradient in this way. Compared with other approaches (such as the algorithms described in Glynn, 1990; Rubinstein, 1991; Williams, 1992, for example), it is especially suitable for systems with large state spaces, when the time between visits to a recurrent state is large but the mixing time of the controlled POMDP is short. However, it can suffer from the problem of high variance in its estimates. We seek to alter GPOMDP so that the estimation variance is reduced, and thereby reduce the number of steps required to train a controller.

One generic approach to reducing the variance of Monte Carlo estimates of integrals is to use an additive control variate (see, for example, Hammersley and Handscomb, 1965; Fishman, 1996; Evans and Swartz, 2000). Suppose we wish to estimate the integral of the function $\mathrm{f} : \mathcal{X} \to \mathbb{R}$, and we happen to know the value of the integral of another function on the same space $\varphi : \mathcal{X} \to \mathbb{R}$. As we have

$$\int_{\mathcal{X}} \mathrm{f}(x) = \int_{\mathcal{X}} (\mathrm{f}(x) - \varphi(x)) + \int_{\mathcal{X}} \varphi(x) \tag{1}$$

the integral of $\mathrm{f}(x) - \varphi(x)$ can be estimated instead. Obviously if $\varphi(x) = \mathrm{f}(x)$ then we have managed to reduce our variance to zero. More generally,

$$\mathrm{Var}(f - \varphi) = \mathrm{Var}(f) - 2\mathrm{Cov}(f, \varphi) + \mathrm{Var}(\varphi).$$

If φ and f are strongly correlated, so that the covariance term on the right hand side is greater than the variance of φ, then a variance improvement has been made over the original estimation problem.

In this paper, we consider two applications of the control variate approach to the problem of gradient estimation in reinforcement learning. The first is the technique of adding a baseline, which is often used as a way to affect estimation variance whilst adding no bias. We show that adding a baseline can be viewed as a control variate method, and we find the optimal choice of baseline to use. We show that the additional variance of a suboptimal baseline can be expressed as a certain weighted squared distance between the baseline and the optimal one. A constant baseline, which

does not depend on the state, has been commonly suggested (Sutton and Barto, 1998; Williams, 1992; Kimura et al., 1995, 1997; Kimura and Kobayashi, 1998b; Marbach and Tsitsiklis, 2001). The expectation over all states of the discounted value of the state has been proposed, and widely used, as a constant baseline, by replacing the reward at each step by the difference between the reward and the average reward. We give bounds on the estimation variance that show that, perhaps surprisingly, this may not be the best choice. Our results are consistent with the experimental observations of Dayan (1990).

The second application of the control variate approach is the use of a value function. The discounted value function is usually not known, and needs to be estimated. Using some fixed, or learnt, value function in place of this estimate can reduce the overall estimation variance. Such *actor-critic methods* have been investigated extensively (Barto et al., 1983; Kimura and Kobayashi, 1998a; Baird, 1999; Sutton et al., 2000; Konda and Tsitsiklis, 2000, 2003). Generally the idea is to minimize some notion of distance between the value function and the true discounted value function, using, for example, TD (Sutton, 1988) or Least-Squares TD (Bradtke and Barto, 1996). In this paper we show that this may not be the best approach: selecting a value function to be equal to the true discounted value function is not always the best choice. Even more surprisingly, we give examples for which the use of a value function that is different from the true discounted value function reduces the variance to zero, for no increase in bias. We consider a value function to be forming part of a control variate, and find a corresponding bound on the expected squared error (that is, including the estimation variance) of the gradient estimate produced in this way.

While the main contribution of this paper is in understanding a variety of ideas in gradient estimation as variance reduction techniques, our results suggest a number of algorithms that could be used to augment the GPOMDP algorithm. We present new algorithms to learn the optimum baseline, and to learn a value function that minimizes the bound on the expected squared error of a gradient estimate, and we describe the results of preliminary experiments, which show that these algorithms give performance improvements.

2. Overview of Paper

Section 3 gives some background information. The POMDP setting and controller are defined, and the measure of performance and its gradient are described. Monte Carlo estimation of integrals, and how these integrals can be estimated, is covered, followed by a discussion of the GPOMDP algorithm, and how it relates to the Monte Carlo estimations. Finally, we outline the control variates that we use.

The samples used in the Monte Carlo estimations are taken from a single sequence of observations. Little can be said about the correlations between these samples. However, Section 4 shows that we can bound the effect they have on the variance in terms of the variance of the iid case (that is, when samples are generated iid according to the stationary distribution of the Markov chain).

Section 5 derives results for a baseline control variate in the iid setting, using results in Section 4 to interpret these as bounds in the more general case. In particular, we give an expression for the minimum variance that may be obtained, and the baseline that achieves this minimum variance. The section also compares the minimum variance against the common technique of using the expectation over states of the discounted value function, and it looks at a restricted class of baselines that use only observation information.

Section 6 looks at the technique of replacing the estimate of the discounted value function with some value function, in a control variate context. It shows that using the true discounted value function may not be the best choice, and that additional gains may be made. It also gives bounds on the expected squared error introduced by a value function.

Section 7 presents an algorithm to learn the optimal baseline. It also presents an algorithm to learn a value function by minimizing an estimate of the resulting expected squared error. Section 8 describes the results of experiments investigating the performance of these algorithms.

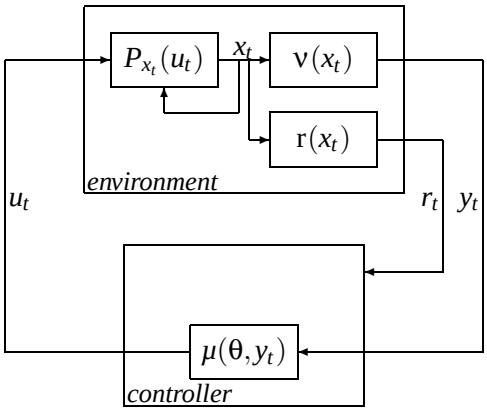
3. Background

Here we formally define the learning setting, including the performance and its gradient. We then give an intuitive discussion of the GPOMDP algorithm, starting with its approximation to the true gradient, and how it may be estimated by Monte Carlo techniques. Finally, we introduce the two variance reduction techniques studied in this paper.

3.1 System Model

A partially observable Markov decision process (POMDP) can be modelled by a system consisting of a state space, $\mathcal{S}$, an action space, $\mathcal{U}$, and an observation space, $\mathcal{Y}$, all of which will be considered finite here. State transitions are governed by a set of probability transition matrices $P(u)$, where $u \in \mathcal{U}$, components of which will be denoted $\mathrm{p}_{ij}(u)$, where $i, j \in \mathcal{S}$. There is also an observation process $\nu : \mathcal{S} \to \mathcal{P}_{\mathcal{Y}}$, where $\mathcal{P}_{\mathcal{Y}}$ is the space of probability distributions over $\mathcal{Y}$, and a reward function $\mathrm{r} : \mathcal{S} \to \mathbb{R}$. Together these define the POMDP $(\mathcal{S}, \mathcal{U}, \mathcal{Y}, P, \nu, \mathrm{r})$.

A policy for this POMDP is a mapping $\mu : \mathcal{Y}^* \to \mathcal{P}_{\mathcal{U}}$, where $\mathcal{Y}^*$ denotes the space of all finite sequences of observations $y_1, \ldots, y_t \in \mathcal{Y}$ and $\mathcal{P}_{\mathcal{U}}$ is the space of probability distributions over $\mathcal{U}$. If only the set of reactive policies $\mu : \mathcal{Y} \to \mathcal{P}_{\mathcal{U}}$ is considered then the joint process of state, observation and action, denoted $\{X_t, Y_t, U_t\}$, is Markov. This paper considers reactive parameterized policies $\mu(y, \theta)$, where $\theta \in \mathbb{R}^K$ and $y \in \mathcal{Y}$. A reactive parameterized policy together with a POMDP defines a *controlled POMDP* $(\mathcal{S}, \mathcal{U}, \mathcal{Y}, P, \nu, \mathrm{r}, \mu)$. See Figure 1.



Average Reward: $\eta = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \mathrm{r}(X_t)\right]$

Figure 1: POMDP with reactive parameterized policy

Given a controlled POMDP the subprocess of states, $\{X_t\}$, is also Markov. A parameterized transition matrix $P(\theta)$, with entries $\mathrm{p}_{ij}(\theta)$, can be constructed, with

$$\mathrm{p}_{ij}(\theta) = \mathbb{E}_{y\sim\nu(i)}\left[\mathbb{E}_{u\sim\mu(y,\theta)}\left[\mathrm{p}_{ij}(u)\right]\right] = \sum_{y\in\mathcal{Y},u\in\mathcal{U}} \nu_y(i)\mu_u(y,\theta)\mathrm{p}_{ij}(u),$$

where $\nu_y(i)$ denotes the probability of observation y given the state i, and $\mu_u(y,\theta)$ denotes the probability of action u given the parameters θ and an observation y. The Markov chain $\mathrm{M}(\theta) = (\mathcal{S}, P(\theta))$ then describes the behavior of the process $\{X_t\}$.

We will also be interested in the special case where the state is fully observable.

Definition 1. *A* controlled Markov decision process *is a controlled POMDP* $(\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,\mathrm{r},\mu)$ *with* $\mathcal{Y}=\mathcal{S}$ *and* $\nu_y(i)=\delta_{yi}$, *where*

$$\delta_{yi} = \begin{cases} 1 & y = i \\ 0 & \textit{otherwise}, \end{cases}$$

and is defined by the tuple $(\mathcal{S},\mathcal{U},P,\mathrm{r},\mu)$.

In this case the set of reactive policies contains the optimal policy, that is, for our performance measure there is a reactive policy that will perform at least as well as any history dependent policy. Indeed, we need only consider mappings to point distributions over actions. Of course, this is not necessarily true of the parameterized class of reactive policies. In the partially observable setting the optimal policy may be history dependent; although a reactive policy may still perform well. For a study of using reactive policies for POMDPs see Singh et al. (1994); Jaakkola et al. (1995); Baird (1999). For a recent survey of POMDP techniques see Aberdeen (2002).

We operate under a number of assumptions for the controlled POMDP $(\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,\mathrm{r},\mu)$. Note that any arbitrary vector v is considered to be a column vector, and that we write v' to denote its transpose, a row vector. Also, the operator ∇ takes a function $\mathrm{f}(\theta)$ to a vector of its partial derivatives, that is

$$\nabla \mathrm{f}(\theta) = \left(\frac{\partial \mathrm{f}(\theta)}{\partial\theta_1},\ldots,\frac{\partial \mathrm{f}(\theta)}{\partial\theta_K}\right)',$$

where θ_k denotes the k^{th} element of θ.

Assumption 1. *For all* $\theta\in\mathbb{R}^K$ *the Markov chain* $\mathrm{M}(\theta)=(\mathcal{S},P(\theta))$ *is irreducible and aperiodic (ergodic), and hence has a unique stationary distribution* $\pi(\theta)$ *satisfying*

$$\pi(\theta)'P(\theta) = \pi(\theta)'$$

The terms *irreducible* and *aperiodic* are defined in Appendix A. Appendix A also contains a discussion of Assumption 1 and how both the irreducibility and aperiodicity conditions may be relaxed.

Assumption 2. *There is a* $\mathbf{R}<\infty$ *such that for all* $i\in\mathcal{S}$, $|\mathrm{r}(i)|\leq\mathbf{R}$.

Assumption 3. *For all* $u\in\mathcal{U}$, $y\in\mathcal{Y}$ *and* $\theta\in\mathbb{R}^K$ *the partial derivatives*

$$\frac{\partial\mu_u(y,\theta)}{\partial\theta_k}, \qquad \forall k\in\{1,\ldots,K\}$$

exist and there is a $\mathbf{B} < \infty$ *such that the Euclidean norms*

$$\left\| \frac{\nabla \mu_u(y,\theta)}{\mu_u(y,\theta)} \right\|$$

are uniformly bounded by $\mathbf{B}$*. We interpret* $0/0$ *to be* 0 *here, that is, we may have* $\mu_u(y,\theta) = 0$ *provided* $\|\nabla \mu_u(y,\theta)\| = 0$*. The Euclidean norm of a vector* v *is given by* $\sqrt{\sum_k v_k^2}$*.*

Note that Assumption 3 implies that

$$\left\| \frac{\nabla \mathrm{p}_{ij}(\theta)}{\mathrm{p}_{ij}(\theta)} \right\| \leq \mathbf{B},$$

where, as in Assumption 3, we interpret $0/0$ to be 0, and so we may have $\mathrm{p}_{ij}(\theta) = 0$ provided $\|\nabla \mathrm{p}_{ij}(\theta)\| = 0$. This bound can be seen from

$$\begin{aligned}
\left\|\nabla \mathrm{p}_{ij}(\theta)\right\| &= \left\| \nabla \sum_{y \in \mathcal{Y}, u \in \mathcal{U}} \nu_y(i) \mu_u(y,\theta) \mathrm{p}_{ij}(u) \right\| \\
&= \left\| \sum_{y \in \mathcal{Y}, u \in \mathcal{U}} \nu_y(i) \nabla \mu_u(y,\theta) \mathrm{p}_{ij}(u) \right\| \\
&\leq \mathbf{B} \sum_{y \in \mathcal{Y}, u \in \mathcal{U}} \nu_y(i) \mu_u(y,\theta) \mathrm{p}_{ij}(u) \\
&= \mathbf{B} \mathrm{p}_{ij}(\theta).
\end{aligned}$$

A useful measure of the system's performance is the expected average reward,

$$\eta(\theta) \stackrel{\text{def}}{=} \lim_{T \to \infty} \mathbb{E}\left[\frac{1}{T} \sum_{t=0}^{T-1} \mathrm{r}(X_t) \right]. \tag{2}$$

From Equation (24) in Appendix A we see that $\eta(\theta) = \mathbb{E}[\mathrm{r}(X) | X \sim \pi(\theta)]$, and hence is independent of the starting state. In this paper we analyze certain training algorithms that aim to select a policy such that this quantity is (locally) maximized.

It is also useful to consider the discounted value function,

$$\mathrm{J}_\beta(i,\theta) \stackrel{\text{def}}{=} \lim_{T \to \infty} \mathbb{E}\left[\sum_{t=0}^{T-1} \beta^t \mathrm{r}(X_t) \middle| X_0 = i \right].$$

Throughout the rest of the paper the dependence upon θ is assumed, and dropped in the notation.

3.2 Gradient Calculation

It is shown in Baxter and Bartlett (2001) that we can calculate an approximation to the gradient of the expected average reward by

$$\nabla_\beta \eta = \sum_{i,j \in \mathcal{S}} \pi_i \nabla p_{ij} \mathrm{J}_\beta(j),$$

and that the limit of $\nabla_\beta \eta$ as β approaches 1 is the true gradient $\nabla\eta$. Note that $\nabla_\beta\eta$ is a parameterized vector in $\mathbb{R}^K$ approximating the gradient of η, and there need not exist any function $f(\theta)$ with $\nabla f(\theta) = \nabla_\beta \eta$.

The gradient approximation $\nabla_\beta\eta$ can be considered as the integration over the state transition space,

$$\nabla_\beta \eta = \int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} J_\beta(j) \mathfrak{C}(di \times dj), \tag{3}$$

where $\mathfrak{C}$ is a counting measure, that is, for a countable space $\mathcal{C}$, and a set $A \subset \mathcal{C}$, we have $\mathfrak{C}(A) = \text{card}(A)$ when A is finite, and $\mathfrak{C}(A) = \infty$ otherwise. Here $\text{card}(A)$ is the cardinality of the set A. It is unlikely that the true value function will be known. The value function can, however, be expressed as the integral over a sample path of the chain, as Assumption 1 implies ergodicity.

$$\nabla_\beta \eta = \int_{(i_0,i_1,\ldots)\in\mathcal{S}\times\mathcal{S}\times\ldots} \pi_{i_0} (\nabla p_{i_0 i_1}) p_{i_1 i_2} p_{i_2 i_3} \ldots \left(r(i_1) + \beta r(i_2) + \beta^2 r(i_3) + \cdots\right) \mathfrak{C}(di_0 \times \ldots).$$

To aid in analysis, the problem will be split into an integral and a sub integral problem.

$$\begin{aligned}
\nabla_\beta \eta &= \int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \int_{(x_1,\ldots)\in\mathcal{S}\times\ldots} \pi_i (\nabla p_{ij}) \delta_{x_1 j} p_{x_1 x_2} \ldots (r(x_1) + \cdots) \mathfrak{C}(dx_1 \times \ldots) \mathfrak{C}(di \times dj) \\
&= \int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i (\nabla p_{ij}) \int_{(x_1,\ldots)\in\mathcal{S}\times\ldots} \delta_{x_1 j} p_{x_1 x_2} \ldots (r(x_1) + \cdots) \mathfrak{C}(dx_1 \times \ldots) \mathfrak{C}(di \times dj).
\end{aligned}$$

3.3 Monte Carlo Estimation

Integrals can be estimated through the use of Monte Carlo techniques by averaging over samples taken from a particular distribution (see Hammersley and Handscomb, 1965; Fishman, 1996; Evans and Swartz, 2000). Take a function $f : \mathcal{X} \to \mathbb{R}$ and a probability distribution ρ over the space $\mathcal{X}$. An unbiased estimate of $\int_{x\in\mathcal{X}} f(x)$ can be generated from samples $\{x_0, x_1, \ldots, x_{m-1}\}$ taken from ρ by

$$\frac{1}{m} \sum_{n=0}^{m-1} \frac{f(x_n)}{\rho(x_n)}.$$

Consider a finite ergodic Markov chain $M = (\mathcal{S}, P)$ with stationary distribution π. Generate the Markov process $\{X_t\}$ from M starting from the stationary distribution. The integral of the function $f : \mathcal{S} \to \mathbb{R}$ over the space $\mathcal{S}$ can be estimated by

$$\frac{1}{T} \sum_{t=0}^{T-1} \frac{f(X_t)}{\pi_{X_t}}.$$

This can be used to estimate the integral

$$\int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} J_\beta(j) \mathfrak{C}(di \times dj).$$

The finite ergodic Markov chain $M = (\mathcal{S}, P)$, with stationary distribution π, can be used to create the extended Markov process $\{X_t, X_{t+1}\}$ and its associated chain. Its stationary distribution has the probability mass function $\rho(i,j) = \pi_i p_{ij}$, allowing the estimation of the above integral by

$$\frac{1}{T} \sum_{t=0}^{T-1} \frac{\nabla p_{X_t X_{t+1}}}{p_{X_t X_{t+1}}} J_{t+1}, \qquad J_t = \sum_{s=t}^{\infty} \beta^{s-t} r(X_s). \tag{4}$$

In addition to the Monte Carlo estimation, the value function has been replaced with an unbiased estimate of the value function. In practice we would need to truncate this sum; a point discussed in the next section. Note, however, that

$$\begin{aligned}\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla p_{X_tX_{t+1}}}{p_{X_tX_{t+1}}}J_{t+1}\right] &= \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\frac{\nabla p_{X_tX_{t+1}}}{p_{X_tX_{t+1}}}\mathbb{E}\left[J_{t+1}\middle|X_{t+1}\right]\right]\\ &= \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla p_{X_tX_{t+1}}}{p_{X_tX_{t+1}}}\mathrm{J}_\beta(X_{t+1})\right].\end{aligned}$$

We will often be looking at estimates produced by larger Markov chains, such as that formed by the process $\{X_t, Y_t, U_t, X_{t+1}\}$. The discussion above also holds for functions on such chains.

3.4 GPOMDP Algorithm

The GPOMDP algorithm uses a single sample path of the Markov process $\{Z_t\} = \{X_t, Y_t, U_t, X_{t+1}\}$ to produce an estimate of $\nabla_\beta \eta$. We denote an estimate produced by GPOMDP with T samples by Δ_T.

$$\Delta_T \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}J_{t+1}, \qquad J_t \stackrel{\text{def}}{=} \sum_{s=t}^{T}\beta^{s-t}\mathrm{r}(X_s). \tag{5}$$

This differs from the estimate given in (4), but can be obtained similarly by considering the estimation of $\nabla_\beta \eta$ by samples from $\{Z_t\}$, and noting that

$$\nabla p_{ij} = \sum_{y\in\mathcal{Y},u\in\mathcal{U}} \nu_y(i)\nabla\mu_u(y)\mathrm{p}_{ij}(u).$$

GPOMDP can be represented as the two dimensional calculation

$$\begin{array}{rccccccccc}\Delta_T = \frac{1}{T}\Big(& \mathrm{f}(Z_0) & J_1 & + & \mathrm{f}(Z_1) & J_2 & + & \cdots & + & \mathrm{f}(Z_{T-1}) \quad J_T \quad \Big)\\ & & \|^{\text{def}} & & & \|^{\text{def}} & & & & \|^{\text{def}}\\ & & \mathrm{g}(Z_0) & & & \mathrm{g}(Z_1) & & \vdots & & \mathrm{g}(Z_{T-1})\\ & & +\beta\mathrm{g}(Z_1) & & & +\beta\mathrm{g}(Z_2) & & & & \\ & & +\beta^2\mathrm{g}(Z_2) & & & \vdots & & & & \\ & & \vdots & & & +\beta^{T-2}\mathrm{g}(Z_{T-1}) & & & & \\ & & +\beta^{T-1}\mathrm{g}(Z_{T-1}) & & & & & & & \end{array}$$

where $\mathrm{f}(Z_t) = (\nabla\mu_{U_t}(Y_t))/\mu_{U_t}(Y_t)$ and $\mathrm{g}(Z_t) = \mathrm{r}(X_{t+1})$.

One way to understand the behavior of GPOMDP is to assume that the chains being used to calculate each J_t sample are independent. This is reasonable when the chain is rapidly mixing and T is large compared with the mixing time, because then most pairs J_{t_1} and J_{t_2} are approximately independent. Replacing J_t by these independent versions, $J_t^{(\text{ind})}$, the calculation becomes

$$\begin{array}{rccccccccc}\Delta_T^{(\text{ind})} \stackrel{\text{def}}{=} \frac{1}{T}\Big(& \mathrm{f}(Z_0) & J_1^{(\text{ind})} & + & \mathrm{f}(Z_1) & J_2^{(\text{ind})} & + & \cdots & + & \mathrm{f}(Z_{T-1}) \quad J_T^{(\text{ind})} \quad \Big)\\ & & \|^{\text{def}} & & & \|^{\text{def}} & & & & \|^{\text{def}}\\ & & \mathrm{g}(Z_{00}) & & & \mathrm{g}(Z_{10}) & & \vdots & & \mathrm{g}\big(Z_{(T-1)0}\big)\\ & & +\beta\mathrm{g}(Z_{01}) & & & +\beta\mathrm{g}(Z_{11}) & & & & \\ & & +\beta^2\mathrm{g}(Z_{02}) & & & \vdots & & & & \\ & & \vdots & & & +\beta^{T-2}\mathrm{g}\big(Z_{1(T-2)}\big) & & & & \\ & & +\beta^{T-1}\mathrm{g}\big(Z_{0(T-1)}\big) & & & & & & & \end{array}$$

where the truncated process $\{Z_{tn}\}$ is an independent sample path generated from the Markov chain of the associated POMDP starting from the state $Z_t = Z_{t0}$.

The truncation of the discounted sum of future rewards would cause a bias from $\nabla_\beta \eta$. By considering T to be large compared to $1/(1-\beta)$ then this bias becomes small for a large proportion of the samples. Replacing each $J_t^{(\mathrm{ind})}$ by an untruncated version, $J_t^{(\mathrm{est})}$, shows how GPOMDP can be thought of as similar to the calculation

$$\begin{array}{ccccccccc}
\Delta_T^{(\mathrm{est})} \stackrel{\mathrm{def}}{=} \frac{1}{T}\Big(& \mathrm{f}(Z_0) & J_1^{(\mathrm{est})} & + & \mathrm{f}(Z_1) & J_2^{(\mathrm{est})} & + \cdots + & \mathrm{f}(Z_{T-1}) & J_T^{(\mathrm{est})} \Big) \\
& & \| & & & \| & & & \| \\
& & \mathrm{g}(Z_{00}) & & & \mathrm{g}(Z_{10}) & \vdots & & \mathrm{g}\big(Z_{(T-1)0}\big) \\
& & +\beta \mathrm{g}(Z_{01}) & & & +\beta \mathrm{g}(Z_{11}) & & & +\beta \mathrm{g}\big(Z_{(T-1)1}\big) \\
& & +\beta^2 \mathrm{g}(Z_{02}) & & & +\beta^2 \mathrm{g}(Z_{12}) & & & +\beta^2 \mathrm{g}\big(Z_{(T-1)2}\big) \\
& & \vdots & & & \vdots & & & \vdots
\end{array}$$

The altered Δ_T sum can be written as

$$\Delta_T^{(\mathrm{est})} = \frac{1}{T} \sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)} J_{t+1}^{(\mathrm{est})}. \tag{6}$$

3.5 Variance Reduction

Equation (1) shows how a control variate can be used to change an estimation problem. To be of benefit the use of the control variate must lower estimation variance, and the integral of the control variate must have a known value. We look at two classes of control variate for which the value of the integral may be determined (or assumed).

The Monte Carlo estimates performed use correlated samples, making it difficult to analyze the variance gain. Given that we wish to deal with quite unrestricted environments, little is known about this sample correlation. We therefore consider the case of iid samples and show how this case gives a bound on the case using correlated samples.

The first form of control variate considered is the baseline control variate. With this, the integral shown in Equation (3) is altered by a control variate of the form $\pi_i \nabla p_{ij} \mathrm{b}(i)$.

$$\begin{aligned}
\int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} \mathrm{J}_\beta(j) \mathfrak{C}(di\times dj) &= \int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} \big(\mathrm{J}_\beta(j) - \mathrm{b}(i)\big) \mathfrak{C}(di\times dj) \\
&\quad + \int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} \mathrm{b}(i) \mathfrak{C}(di\times dj)
\end{aligned}$$

The integral of the control variate term is zero, since

$$\begin{aligned}
\int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} \mathrm{b}(i) \mathfrak{C}(di\times dj) &= \sum_{i\in\mathcal{S}} \pi_i \mathrm{b}(i) \nabla \sum_{j\in\mathcal{S}} p_{ij} \\
&= \sum_{i\in\mathcal{S}} \pi_i \mathrm{b}(i) \nabla(1) \\
&= 0.
\end{aligned} \tag{7}$$

Thus, we are free to select an arbitrary $\mathrm{b}(i)$ with consideration for the variance minimization alone.

The second form of control variate considered is constructed from a value function, $V(j)$, a mapping $\mathcal{S} \to \mathbb{R}$.

$$\begin{aligned}\int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} J_\beta(j) \mathfrak{C}(di \times dj) &= \int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} \left(J_\beta(j) - \left(J_\beta(j) - V(j)\right)\right) \mathfrak{C}(di \times dj) \\ &\quad + \int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \pi_i \nabla p_{ij} \left(J_\beta(j) - V(j)\right) \mathfrak{C}(di \times dj)\end{aligned}$$

The integral of this control variate (the last term in the equation above) is the error associated with using a value function in place of the true discounted value function. The task is then to find a value function such that the integral of the control variate is small, and yet it still provides good variance minimization of the estimated integral.

Note that the integrals being estimated here are vector quantities. We consider the trace of the covariance matrix of these quantities, that is, the sum of the variance of the components of the vector. Given the random vector $A = (A_1, A_2, \ldots, A_k)'$, we write

$$\mathrm{Var}(A) = \sum_{m=1}^{k} \mathrm{Var}(A_m) = \mathbb{E}\left[(A - \mathbb{E}[A])'(A - \mathbb{E}[A])\right] = \mathbb{E}\left[(A - \mathbb{E}[A])^2\right],$$

where, for a vector a, a^2 denotes $a'a$.

4. Dependent Samples

In Sections 5 and 6 we study the variance of quantities that, like $\Delta_T^{(\mathrm{est})}$ (Equation (6)), are formed from the sample average of a process generated by a controlled (PO)MDP. From Section 3 we know this process is Markov, is ergodic, and has a stationary distribution, and so the sample average is an estimate of the expectation of a sample drawn from the stationary distribution, π (note that, as in Section 3.3, we can also look at samples formed from an extended space, and its associated stationary distributions). In this section we investigate how the variance of the sample average relates to the variance of a sample drawn from π. This allows us to derive results for the variance of a sample drawn from π and relate them to the variance of the sample average. In the iid case, that is, when the process generates a sequence of samples $X_0, \ldots, X_{T-1}$ drawn independently from the distribution π, we have the relationship

$$\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} f(X_t)\right) = \frac{1}{T}\mathrm{Var}(f(X)),$$

where X is a random variable also distributed according to π. More generally, however, correlation between the samples makes finding an exact relationship difficult. Instead we look to find a bound of the form

$$\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} f(X_t)\right) \leq h\left(\frac{1}{T}\mathrm{Var}(f(X))\right),$$

where h is some "well behaved" function.

We first define a notion of mixing time for a Markov chain. The mixing time is a measure of the forgetfulness of a Markov chain. More specifically, it is a measure of how long it takes for the distance between the distributions of two sequences, starting in distinct states, to become small. The distance measure we use is the total variation distance.

Definition 2. *The* total variation distance *between two distributions p,q on the finite set $\mathcal{S}$ is given by*

$$\mathrm{d}_{TV}(p,q) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{i\in\mathcal{S}} |p_i - q_i|.$$

Definition 3. *The* mixing time *of a finite ergodic Markov chain $M=(\mathcal{S},P)$ is defined as*

$$\tau \stackrel{\text{def}}{=} \min\left\{ t>0 : \max_{i,j} \mathrm{d}_{TV}\left(P_i^t, P_j^t\right) \le e^{-1} \right\},$$

where P_i^t denotes the i^{th} row of the t-step transition matrix P^t.

The results in this section are given for a Markov chain with mixing time τ. In later sections we will use τ as a measure of the mixing time of the resultant Markov chain of states of a controlled (PO)MDP, but will look at sample averages over larger spaces. The following lemma, due to Bartlett and Baxter (2002), shows that the mixing time does not grow too fast when looking at the Markov chain on sequences of states.

Lemma 1. *(Bartlett and Baxter, 2002, Lemma 4.3) If the Markov chain $M=(\mathcal{S},P)$ has mixing time τ, then the Markov chain formed by the process $\{X_t, X_{t+1}, \ldots, X_{t+k}\}$ has mixing time $\tilde{\tau}$, where*

$$\tilde{\tau} \le \tau \ln\left(e(k+1)\right).$$

Note 1. *For a controlled POMDP, the Markov chain formed by the process $\{X_t, X_{t+1}, \ldots, X_{t+k}\}$ has the same mixing time as the Markov chain formed by the process $\{X_t, Y_t, U_t, X_{t+1}, \ldots, Y_{t+k-1}, U_{t+k-1}, X_{t+k}\}$.*

We now look at showing the relationship between the covariance between two samples in a sequence and the variance of an individual sample. We show that the gain of the covariance of two samples X_t, X_{t+s} over the variance of an individual sample decreases exponentially in s.

Theorem 2. *Let $M=(\mathcal{S},P)$ be a finite ergodic Markov chain, and let π be its stationary distribution. Let f be some mapping $\mathrm{f}:\mathcal{S}\to\mathbb{R}$. The tuple (M,f) has associated positive constants α and $\mathbf{L}$ (called* mixing constants *$(\alpha,\mathbf{L})$) such that, for all $t\ge 0$,*

$$|\mathrm{Cov}_\pi(t;\mathrm{f})| \le \mathbf{L}\alpha^t \mathrm{Var}(\mathrm{f}(X))$$

where $X\sim\pi$, and $\mathrm{Cov}_\pi(t;\mathrm{f})$ is the auto-covariance of the process $\{\mathrm{f}(X_s)\}$, i.e. $\mathrm{Cov}_\pi(t;\mathrm{f}) = \mathbb{E}_\pi\left[(\mathrm{f}(X_s)-\mathbb{E}_\pi\mathrm{f}(X_s))\,(\mathrm{f}(X_{s+t})-\mathbb{E}_\pi\mathrm{f}(X_{s+t}))\right]$, where $\mathbb{E}_\pi[\cdot]$ denotes the expectation over the chain with initial distribution π. Furthermore, if M has mixing time τ, we have:

1. *for reversible M, and any f, we may choose $\mathbf{L}=2e$ and $\alpha=\exp(-1/\tau)$; and*
2. *for any M (that is, any finite ergodic M), and any f, we may choose $\mathbf{L}=\sqrt{2|\mathcal{S}|e}$ and $\alpha=\exp(-1/(2\tau))$.*

The proof is shown in Appendix B, along with proofs for the rest of this section.
Using this result, the variance of the sample average can be bounded as follows.

Theorem 3. *Let $M = (\mathcal{S}, P)$ be a finite ergodic Markov chain, with mixing time τ, and let π be its stationary distribution. Let f be some mapping $\mathrm{f} : \mathcal{S} \to \mathbb{R}$. Let $\{X_t\}$ be a sample path generated by M, with initial distribution π, and let $X \sim \pi$. With (M, f) mixing constants $(\alpha, \mathbf{L})$ chosen such that $\alpha \leq \exp(-1/(2\tau))$, there is an $\Omega^* \leq 6\mathbf{L}\tau$ such that*

$$\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\right) \leq \frac{\Omega^*}{T}\mathrm{Var}(\mathrm{f}(X)).$$

Provided acceptable mixing constants can be chosen, Theorem 3 gives the same rate as in the case of independent random variables, that is, the variance decreases as $O(1/T)$. The most that can be done to improve the bound of Theorem 3 is to reduce the constant Ω^*. It was seen, in Theorem 2, that good mixing constants can be chosen for functions on reversible Markov chains. We would like to deal with more general chains also, and the mixing constants given in Theorem 2 for functions on ergodic Markov chains lead to Ω^* increasing with the size of the state space. However, for bounded functions on ergodic Markov chains we have the following result:

Theorem 4. *Let $M = (\mathcal{S}, P)$ be a finite ergodic Markov chain, and let π be its stationary distribution. If M has mixing time τ, then for any function $\mathrm{f} : \mathcal{S} \to [-c, c]$ and any $0 < \varepsilon < e^{-1}$, we have*

$$\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\right) \leq \varepsilon + \left(1 + 25\tau(1+c)\varepsilon + 4\tau\ln\frac{1}{\varepsilon}\right)\frac{1}{T}\mathrm{Var}\left(\mathrm{f}(X)\right),$$

where $\{X_t\}$ is a process generated by M with initial distribution $X_0 \sim \pi$, and $X \sim \pi$.

Here we have an additional error ε, which we may decrease at the cost of a $\ln\varepsilon^{-1}$ penalty in the constant multiplying the variance term.

Consider the following corollary of Theorem 4.

Corollary 5. *Let $M = (\mathcal{S}, P)$ be a finite ergodic Markov chain, and let π be its stationary distribution. If M has mixing time τ, then for any function $\mathrm{f} : \mathcal{S} \to [-c, c]$, we have*

$$\begin{aligned}\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\right) \quad \leq \quad & 4\tau\ln\left(7(1+c) + \frac{1}{4\tau}\left(\frac{1}{T}\mathrm{Var}\left(\mathrm{f}(X)\right)\right)^{-1}\right)\frac{1}{T}\mathrm{Var}\left(\mathrm{f}(X)\right) \\ & + (1+8\tau)\frac{1}{T}\mathrm{Var}\left(\mathrm{f}(X)\right)\end{aligned}$$

where $\{X_t\}$ is a process generated by M with initial distribution $X_0 \sim \pi$, and $X \sim \pi$.

Here, again, our bound approaches zero as $\mathrm{Var}(\mathrm{f}(X))/T \to 0$, but at the slightly slower rate of

$$O\left(\frac{1}{T}\mathrm{Var}(\mathrm{f}(X))\ln\left(e + \left(\frac{1}{T}\mathrm{Var}(\mathrm{f}(X))\right)^{-1}\right)\right),$$

where we have ignored the dependence on τ and c. For a fixed variance the rate of decrease in T is $O(\ln(T)/T)$, slightly worse than the $O(1/T)$ rate for independent random variables.

5. Baseline Control Variate

As stated previously, a baseline may be selected with regard given only to the estimation variance. In this section we consider how the baseline affects the variance of our gradient estimates when the samples are iid, and the discounted value function is known. We show that, when using Theorem 3 or Theorem 4 to bound covariance terms, this is reasonable, and in fact the error in analysis (that is, from not analyzing the variance of Δ_T with baseline directly) associated with the choice of baseline is negligible. This statement will be made more precise later.

Section 5.2 looks at the Markov chain of states generated by the controlled POMDP and is concerned with producing a baseline $\mathrm{b}_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}$ to minimize the variance

$$\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}}) = \mathrm{Var}_\pi\left(\frac{\nabla p_{ij}}{p_{ij}}\left(\mathrm{J}_\beta(j) - \mathrm{b}_{\mathcal{S}}(i)\right)\right), \tag{8}$$

where, for some $f : \mathcal{S}\times\mathcal{S} \to \mathbb{R}^K$, $\mathrm{Var}_\pi(\mathrm{f}(i,j)) = \mathbb{E}_\pi\left(\mathrm{f}(i,j) - \mathbb{E}_\pi \mathrm{f}(i,j)\right)^2$ with $\mathbb{E}_\pi[\cdot]$ denoting the expectation over the random variables i, j with $i \sim \pi$ and $j \sim P_i$. Equation (8) serves as a definition of $\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}})$. The section gives the minimal value of this variance, and the minimizing baseline. Additionally, the minimum variance and corresponding baseline is given for the case where the baseline is a constant, $b \in \mathbb{R}$. In both cases, we give expressions for the excess variance of a suboptimal baseline, in terms of a weighted squared distance between the baseline and the optimal one. We can thus show the difference between the variance for the optimal constant baseline and the variance obtained when $b = \mathbb{E}_\pi \mathrm{J}_\beta(i)$.

Section 5.3 considers a baseline $\mathrm{b}_{\mathcal{Y}} : \mathcal{Y} \to \mathbb{R}$ for the GPOMDP estimates. It shows how to minimize the variance of the estimate

$$\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \mathrm{Var}_\pi\left(\frac{\nabla \mu_u(y)}{\mu_u(y)}\left(\mathrm{J}_\beta(j) - \mathrm{b}_{\mathcal{Y}}(y)\right)\right), \tag{9}$$

where, for some $f : \mathcal{S}\times\mathcal{Y}\times\mathcal{U}\times\mathcal{S} \to \mathbb{R}^K$, $\mathrm{Var}_\pi(\mathrm{f}(i,y,u,j)) = \mathbb{E}_\pi\left(\mathrm{f}(i,y,u,j) - \mathbb{E}_\pi \mathrm{f}(i,y,u,j)\right)^2$ with, in this case, $\mathbb{E}_\pi[\cdot]$ denoting the expectation over the random variables i, y, u, j with $i \sim \pi$, $y \sim \nu(i)$, $u \sim \mu(y)$, and $j \sim P_i(u)$. Equation (9) serves as a definition of $\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}})$. The case where the state space is fully observed is shown as a consequence.

5.1 Matching Analysis and Algorithm

The analysis in following sections will look at Equation (8) and Equation (9). Here we will show that the results of that analysis can be applied to the variance of a realizable algorithm for generating $\nabla_\beta \eta$ estimates. Specifically, we compare the variance quantity of Equation (9) to a slight variation of the Δ_T estimate produced by GPOMDP, where the chain is run for an extra S steps. We consider the estimate

$$\Delta_T^{(+S)} \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)} J_{t+1}^{(+S)}, \qquad J_t^{(+S)} \stackrel{\text{def}}{=} \sum_{s=t}^{T+S} \beta^{s-t}\mathrm{r}(X_s), \tag{10}$$

and are interested in improving the variance by use of a baseline, that is, by using the estimate

$$\Delta_T^{(+S)}(\mathrm{b}_{\mathcal{Y}}) \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)} - \mathrm{b}_{\mathcal{Y}}(Y_t)\right).$$

We delay the main result of the section, Theorem 7, to gain an insight into the ideas behind it. In Section 3.4 we saw how GPOMDP can be thought of as similar to the estimate $\Delta_T^{(\text{est})}$, Equation (6). Using a baseline gives us the new estimate

$$\Delta_T^{(\text{est})}(\mathrm{b}_{\mathcal{Y}}) \stackrel{\text{def}}{=} \frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(\text{est})} - \mathrm{b}_{\mathcal{Y}}(Y_t)\right). \tag{11}$$

The term $J_t^{(\text{est})}$ in Equation (11) is an unbiased estimate of the discounted value function. The following lemma shows that, in analysis of the baseline, we can consider the discounted value function to be known, not estimated.

Lemma 6. *Let $\{X_t\}$ be a random process over the space $\mathcal{X}$. Define arbitrary functions on the space $\mathcal{X}$: $\mathrm{f}:\mathcal{X}\to\mathbb{R}$, $\mathrm{J}:\mathcal{X}\to\mathbb{R}$, and $\mathrm{a}:\mathcal{X}\to\mathbb{R}$. For all t let J_t be a random variable such that $\mathbb{E}[J_t|X_t=i]=\mathrm{J}(i)$. Then*

$$\begin{aligned}\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(J_t-\mathrm{a}(X_t)\right)\right) - \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(\mathrm{J}(X_t)-\mathrm{a}(X_t)\right)\right)\\ = \mathbb{E}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(J_t-\mathrm{J}(X_t)\right)\right)^2\end{aligned}$$

The proof of Lemma 6 is given in Appendix C, along with the proof of Theorem 7 below.

Direct application of Lemma 6 gives,

$$\begin{aligned}\mathrm{Var}\left(\Delta_T^{(\text{est})}(\mathrm{b}_{\mathcal{Y}})\right) &= \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(\mathrm{J}_\beta(X_{t+1})-\mathrm{b}_{\mathcal{Y}}(Y_t)\right)\right)\\ &\quad+\mathbb{E}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(\text{est})}-\mathrm{J}_\beta(X_{t+1})\right)\right)^2.\end{aligned}$$

Thus, we see that we can split the variance of this estimate into two components: the first is the variance of this estimate with $J_t^{(\text{est})}$ replaced by the true discounted value function; and the second is a component independent of our choice of baseline. We can now use Theorem 3 or Corollary 5 to bound the covariance terms, leaving us to analyze Equation (9).

We can obtain the same sort of result, using the same reasoning, for the estimate we are interested in studying in practice: $\Delta_T^{(+S)}(\mathrm{b}_{\mathcal{Y}})$ (see Equation (12) below).

Theorem 7. *Let $D=(\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,\mathrm{r},\mu)$ be a controlled POMDP satisfying Assumptions 1, 2 and 3. Let $M=(\mathcal{S},P)$ be the resultant Markov chain of states, and let π be its stationary distribution; M has a mixing time τ; $\{Z_t\}=\{X_t,Y_t,U_t,X_{t+1}\}$ is a process generated by D, starting $X_0\sim\pi$. Suppose that $\mathrm{a}(\cdot)$ is a function uniformly bounded by $\mathbf{M}$, and $\mathcal{J}(j)$ is the random variable $\sum_{s=0}^{\infty}\beta^s\mathrm{r}(W_s)$ where the states W_s are generated by D starting in $W_0=j$. There are constants $C_1\leq 7+7\mathbf{B}(\mathbf{R}+\mathbf{M})$ and*

$C_2 = 20\tau\mathbf{B}^2\mathbf{R}(\mathbf{R}+\mathbf{M})$ *such that for all* $T, S \geq 1$ *we have*

$$\begin{aligned}
&\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)}-\mathrm{a}(Z_t)\right)\right)\\
&\leq \mathrm{h}\left(\frac{\tau\ln(e(S+1))}{T}\mathrm{Var}_\pi\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\left(\mathrm{J}_\beta(j)-\mathrm{a}(i,y,u,j)\right)\right)\right)\\
&\qquad+\mathrm{h}\left(\frac{\tau\ln(e(S+1))}{T}\mathbb{E}_\pi\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\left(\mathcal{J}(j)-\mathrm{J}_\beta(j)\right)\right)^2\right)\\
&\qquad+\frac{2C_2}{(1-\beta)^2}\left[\ln\frac{1}{\beta}+\ln\left(\frac{C_1}{1-\beta}+\frac{K(1-\beta)^2}{C_2}\right)\right]\frac{(T+S)\ln(e(S+1))}{T}\beta^S,
\end{aligned}$$

where $\mathrm{h}:\mathbb{R}^+\to\mathbb{R}^+$ *is continuous and increasing with* $\mathrm{h}(0)=0$, *and is given by*

$$\mathrm{h}(x) = 9x + 4x\ln\left(\frac{C_1}{1-\beta}+\frac{K}{4}x^{-1}\right).$$

By selecting $S=T$ in Theorem 7, and applying to $\Delta_T^{(+S)}(\mathrm{b}_\mathcal{Y})$ with absolutely bounded $\mathrm{b}_\mathcal{Y}$, we obtain the desired result:

$$\mathrm{Var}\left(\Delta_T^{(+T)}(\mathrm{b}_\mathcal{Y})\right) \leq \mathrm{h}\left(\frac{\tau\ln(e(T+1))}{T}\sigma^2_\mathcal{Y}(\mathrm{b}_\mathcal{Y})\right)+\mathrm{N}(D,T)+O\left(\ln(T)\beta^T\right). \tag{12}$$

Here $\mathrm{N}(D,T)$ is the noise term due to using an estimate in place of the discounted value function, and does not depend on the choice of baseline. The remaining term is of the order $\ln(T)\beta^T$; it is almost exponentially decreasing in T, and hence negligible. The function h is due to the application of Theorem 4, and consequently the discussion in Section 4 on the rate of decrease applies here, that is, a log penalty is paid. In this case, for $\sigma^2_\mathcal{Y}(\mathrm{b}_\mathcal{Y})$ fixed, the rate of decrease is $O(\ln^2(T)/T)$.

Note that we may replace $(\nabla\mu_u(y))/\mu_u(y)$ with $(\nabla p_{ij})/p_{ij}$ in Theorem 7. So if the $(\nabla p_{ij})/p_{ij}$ can be calculated, then Theorem 7 also relates the analysis of Equation 8 with a realizable algorithm for generating $\nabla_\beta\eta$ estimates; in this case an estimate produced by watching the Markov process of states.

5.2 Markov Chains

Here we look at baselines for $\nabla_\beta\eta$ estimates for a parameterized Markov chain and associated reward function (a Markov reward process). The Markov chain of states generated by a controlled POMDP (together with the POMDPs reward function) is an example of such a process. However, the baselines discussed in this section require knowledge of the state to use, and knowledge of $(\nabla\mathrm{p}_{ij}(\theta))/\mathrm{p}_{ij}(\theta)$ to estimate. More practical results for POMDPs are given in the next section.

Consider the following assumption.

Assumption 4. *The parameterized Markov chain* $M(\theta)=(\mathcal{S},P(\theta))$ *and associated reward function* $r:\mathcal{S}\to\mathbb{R}$ *satisfy:* $M(\theta)$ *is irreducible and aperiodic, with stationary distribution* π*; there is a* $\mathbf{R}<\infty$ *such that for all* $i\in\mathcal{S}$ *we have* $|r(i)|\leq\mathbf{R}$*; and for all* $i,j\in\mathcal{S}$, *and all* $\theta\in\mathbb{R}^K$, *the partial derivatives* $\nabla\mathrm{p}_{ij}(\theta)$ *exist, and there is a* $\mathbf{B}<\infty$ *such that* $\|(\nabla\mathrm{p}_{ij}(\theta))/\mathrm{p}_{ij}(\theta)\|\leq\mathbf{B}$.

For any controlled POMDP satisfying Assumptions 1, 2 and 3, Assumption 4 is satisfied for the Markov chain formed by the subprocess $\{X_t\}$ together with the reward function for the controlled POMDP.

Now consider a control variate of the form

$$\varphi_{\mathcal{S}}(i,j) \stackrel{\text{def}}{=} \pi_i \nabla p_{ij} \mathrm{b}_{\mathcal{S}}(i)$$

for estimation of the integral in Equation (3). We refer to the function $\mathrm{b}_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}$ as a baseline.

As shown in Section 3.5, the integral of the baseline control variate $\varphi_{\mathcal{S}}(i,j)$ over $\mathcal{S} \times \mathcal{S}$ can be calculated analytically and is equal to zero. Thus an estimate of the integral

$$\int_{(i,j)\in\mathcal{S}\times\mathcal{S}} \left(\pi_i \nabla p_{ij} \mathrm{J}_\beta(j) - \varphi_{\mathcal{S}}(i,j)\right) \mathfrak{C}(di \times dj)$$

forms an unbiased estimate of $\nabla_\beta \eta$.

The following theorem gives the minimum variance, and the baseline to achieve the minimum variance. We use $\sigma^2_{\mathcal{S}}$ to denote the variance of the estimate without a baseline,

$$\sigma^2_{\mathcal{S}} = \mathrm{Var}_\pi\left(\frac{\nabla p_{ij}}{p_{ij}} \mathrm{J}_\beta(j)\right),$$

and we recall, from Equation (8), that $\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}})$ denotes the variance with a baseline,

$$\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}}) = \mathrm{Var}_\pi\left(\frac{\nabla p_{ij}}{p_{ij}} \left(\mathrm{J}_\beta(j) - \mathrm{b}_{\mathcal{S}}(i)\right)\right).$$

Theorem 8. *Let $M(\theta) = (\mathcal{S}, P(\theta))$ and $r : \mathcal{S} \to \mathbb{R}$ be a parameterized Markov chain and reward function satisfying Assumption 4. Then*

$$\sigma^2_{\mathcal{S}}(\mathrm{b}^*_{\mathcal{S}}) \stackrel{\text{def}}{=} \inf_{\mathrm{b}_{\mathcal{S}} \in \mathbb{R}^{\mathcal{S}}} \sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}}) = \sigma^2_{\mathcal{S}} - \mathbb{E}_{i\sim\pi}\left[\frac{\left(\mathbb{E}\left[(\nabla p_{ij}/p_{ij})^2 \mathrm{J}_\beta(j) \middle| i\right]\right)^2}{\mathbb{E}\left[(\nabla p_{ij}/p_{ij})^2 \middle| i\right]}\right],$$

where $\mathbb{E}[\cdot|i]$ is the expectation over the resultant state j conditioned on being in state i, that is, $j \sim P_i$, and $\mathbb{R}^{\mathcal{S}}$ is the space of functions mapping $\mathcal{S}$ to $\mathbb{R}$. This infimum is attained with the baseline

$$\mathrm{b}^*_{\mathcal{S}}(i) = \frac{\mathbb{E}\left[(\nabla p_{ij}/p_{ij})^2 \mathrm{J}_\beta(j) \middle| i\right]}{\mathbb{E}\left[(\nabla p_{ij}/p_{ij})^2 \middle| i\right]}.$$

The proof uses the following lemma.

Lemma 9. *For any $\mathrm{b}_{\mathcal{S}}$,*

$$\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}}) = \sigma^2_{\mathcal{S}} + \mathbb{E}_\pi\left[\mathrm{b}^2_{\mathcal{S}}(i)\,\mathbb{E}\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2 \middle| i\right] - 2\mathrm{b}_{\mathcal{S}}(i)\mathbb{E}\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2 \mathrm{J}_\beta(j) \middle| i\right]\right].$$

Proof.

$$\begin{aligned}
\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}}) &= \mathbb{E}_\pi\left(\frac{\nabla p_{ij}}{p_{ij}}\left(\mathrm{J}_\beta(j)-\mathrm{b}_{\mathcal{S}}(i)\right)-\mathbb{E}_\pi\left[\frac{\nabla p_{ij}}{p_{ij}}\left(\mathrm{J}_\beta(j)-\mathrm{b}_{\mathcal{S}}(i)\right)\right]\right)^2 \\
&= \mathbb{E}_\pi\left(\left(\frac{\nabla p_{ij}}{p_{ij}}\mathrm{J}_\beta(j)-\mathbb{E}_\pi\left[\frac{\nabla p_{ij}}{p_{ij}}\mathrm{J}_\beta(j)\right]\right)-\left(\frac{\nabla p_{ij}}{p_{ij}}\mathrm{b}_{\mathcal{S}}(i)-\mathbb{E}_\pi\left[\frac{\nabla p_{ij}}{p_{ij}}\mathrm{b}_{\mathcal{S}}(i)\right]\right)\right)^2 \\
&= \sigma^2_{\mathcal{S}}+\mathbb{E}_\pi\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\mathrm{b}_{\mathcal{S}}(i)\right)^2-2\left(\frac{\nabla p_{ij}}{p_{ij}}\mathrm{b}_{\mathcal{S}}(i)\right)'\left(\frac{\nabla p_{ij}}{p_{ij}}\mathrm{J}_\beta(j)\right)\right] \qquad (13) \\
&= \sigma^2_{\mathcal{S}}+\mathbb{E}_{i\sim\pi}\left[\mathrm{b}^2_{\mathcal{S}}(i)\,\mathbb{E}\left[\left(\frac{\nabla p_{\tilde{i}\blacksquare}}{p_{\tilde{i}\blacksquare}}\right)^2\middle|\tilde{\imath}=i\right]\right. \\
&\qquad \left.-2\mathrm{b}_{\mathcal{S}}(i)\mathbb{E}\left[\left(\frac{\nabla p_{\tilde{i}\blacksquare}}{p_{\tilde{i}\blacksquare}}\right)^2\mathrm{J}_\beta(\blacksquare)\middle|\tilde{\imath}=i\right]\right],
\end{aligned}$$

where Equation (13) uses

$$\mathbb{E}_\pi\left[\frac{\nabla p_{ij}}{p_{ij}}\mathrm{b}_{\mathcal{S}}(i)\right]=\int_{(i,j)\in\mathcal{S}\times\mathcal{S}}\pi_i\nabla p_{ij}\mathrm{b}_{\mathcal{S}}(i)\mathfrak{C}(di\times dj)=0,$$

from (7). ■

Proof of Theorem 8. We use Lemma 9 and minimize for each $i\in\mathcal{S}$. Differentiating with respect to each $\mathrm{b}_{\mathcal{S}}(i)$ gives

$$\begin{aligned}
2\mathrm{b}_{\mathcal{S}}(i)\mathbb{E}\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2\middle|i\right]-2\mathbb{E}\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2\mathrm{J}_\beta(j)\middle|i\right] &= 0 \\
\Rightarrow \mathrm{b}_{\mathcal{S}}(i) &= \frac{\mathbb{E}\left[(\nabla p_{ij}/p_{ij})^2\mathrm{J}_\beta(j)\middle|i\right]}{\mathbb{E}\left[(\nabla p_{ij}/p_{ij})^2\middle|i\right]},
\end{aligned}$$

which implies the result. ■

The following theorem shows that the excess variance due to a suboptimal baseline function can be expressed as a weighted squared distance to the optimal baseline.

Theorem 10. *Let $M(\theta)=(\mathcal{S},P(\theta))$ and $r:\mathcal{S}\to\mathbb{R}$ be a parameterized Markov chain and reward function satisfying Assumption 4. Then*

$$\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}})-\sigma^2_{\mathcal{S}}(\mathrm{b}^*_{\mathcal{S}})=\mathbb{E}_\pi\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2\left(\mathrm{b}_{\mathcal{S}}(i)-\mathrm{b}^*_{\mathcal{S}}(i)\right)^2\right].$$

Proof. For each $i\in\mathcal{S}$, define S_i and W_i as

$$\begin{aligned}
S_i &= \mathbb{E}\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2\middle|i\right], \\
W_i &= \mathbb{E}\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2\mathrm{J}_\beta(j)\middle|i\right].
\end{aligned}$$

Lemma 9 and the definition of $\mathrm{b}^*_{\mathcal{S}}$ in Theorem 8 imply that

$$\begin{aligned}
\sigma^2_{\mathcal{S}}(\mathrm{b}_{\mathcal{S}}) - \sigma^2_{\mathcal{S}}(\mathrm{b}^*_{\mathcal{S}}) &= \mathbb{E}_\pi\left[\mathrm{b}^2_{\mathcal{S}}(i)\,S_i - 2\mathrm{b}_{\mathcal{S}}(i)W_i + \frac{W_i^2}{S_i}\right] \\
&= \mathbb{E}_\pi\left(\mathrm{b}_{\mathcal{S}}(i)\sqrt{S_i} - \frac{W_i}{\sqrt{S_i}}\right)^2 \\
&= \mathbb{E}_\pi\left[\left(\mathrm{b}_{\mathcal{S}}(i) - \mathrm{b}^*_{\mathcal{S}}(i)\right)^2 S_i\right] \\
&= \mathbb{E}_\pi\left[\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2 \left(\mathrm{b}_{\mathcal{S}}(i) - \mathrm{b}^*_{\mathcal{S}}(i)\right)^2\right].
\end{aligned}$$

■

The following theorem gives the minimum variance, the baseline to achieve the minimum variance, and the additional variance away from this minimum, when restricted to a constant baseline, $b \in \mathbb{R}$. We use $\sigma^2_{\mathcal{S}}(b)$ to denote the variance with constant baseline b,

$$\sigma^2_{\mathcal{S}}(b) = \mathrm{Var}_\pi\left(\frac{\nabla p_{ij}}{p_{ij}}\left(\mathrm{J}_\beta(j) - b\right)\right). \tag{14}$$

The proof uses Lemma 9 in the same way as the proof of Theorem 8. The proof of the last statement follows that of Theorem 10 by replacing S_i with $S = \mathbb{E}_\pi S_i$, and W_i with $W = \mathbb{E}_\pi W_i$.

Theorem 11. *Let $M(\theta) = (\mathcal{S}, P(\theta))$ and $r : \mathcal{S} \to \mathbb{R}$ be a parameterized Markov chain and reward function satisfying Assumption 4. Then*

$$\sigma^2_{\mathcal{S}}(b^*) \stackrel{\text{def}}{=} \inf_{b\in\mathbb{R}} \sigma^2_{\mathcal{S}}(b) = \sigma^2_{\mathcal{S}} - \frac{\left(\mathbb{E}_\pi\left[(\nabla p_{ij}/p_{ij})^2\,\mathrm{J}_\beta(j)\right]\right)^2}{\mathbb{E}_\pi\left(\nabla p_{ij}/p_{ij}\right)^2}.$$

This infimum is attained with

$$b^* = \frac{\mathbb{E}_\pi\left[(\nabla p_{ij}/p_{ij})^2\,\mathrm{J}_\beta(j)\right]}{\mathbb{E}_\pi\left(\nabla p_{ij}/p_{ij}\right)^2}.$$

The excess variance due to a suboptimal constant baseline b is given by,

$$\sigma^2_{\mathcal{S}}(b) - \sigma^2_{\mathcal{S}}(b^*) = \mathbb{E}_\pi\left(\frac{\nabla p_{ij}}{p_{ij}}\right)^2 (b - b^*)^2.$$

A baseline of the form $b = \mathbb{E}_\pi \mathrm{J}_\beta(i)$ is often promoted as a good choice. Theorem 11 gives us a tool to measure how far this choice is from the optimum.

Corollary 12. *Let $M(\theta) = (\mathcal{S}, P(\theta))$ and $r : \mathcal{S} \to \mathbb{R}$ be a Markov chain and reward function satisfying Assumption 4. Then*

$$\sigma^2_{\mathcal{S}}(\mathbb{E}\mathrm{J}_\beta(i)) - \sigma^2_{\mathcal{S}}(b^*) = \frac{\left(\mathbb{E}_\pi\left(\nabla p_{ij}/p_{ij}\right)^2 \mathbb{E}_\pi \mathrm{J}_\beta(j) - \mathbb{E}_\pi\left[(\nabla p_{ij}/p_{ij})^2\,\mathrm{J}_\beta(j)\right]\right)^2}{\mathbb{E}_\pi\left(\nabla p_{ij}/p_{ij}\right)^2}.$$

Notice that the sub-optimality of the choice $b = \mathbb{E}_\pi \mathrm{J}_\beta(i)$ depends on the independence of the random variables $(\nabla p_{ij}/p_{ij})^2$ and $\mathrm{J}_\beta(j)$; if they are nearly independent, $\mathbb{E}_\pi \mathrm{J}_\beta(i)$ is a good choice.

Of course, when considering sample paths of Markov chains, Corollary 12 only shows the difference of the two *bounds* on the variance given by Theorem 7, but it gives an indication of the true distance. In particular, as the ratio of the mixing time to the sample path length becomes small, the difference between the variances in the dependent case approaches that of Corollary 12.

5.3 POMDPs

Consider a control variate over the extended space $\mathcal{S} \times \mathcal{Y} \times \mathcal{U} \times \mathcal{S}$ of the form

$$\varphi(i,y,u,j) = \pi_i \nu_y(i) \nabla \mu_u(y) \mathrm{p}_{ij}(u) \mathrm{b}(i,y).$$

Again, its integral is zero.

$$\begin{aligned}&\int_{(i,y,u,j)\in\mathcal{S}\times\mathcal{Y}\times\mathcal{U}\times\mathcal{S}} \varphi(i,y,u,j)\mathfrak{C}(di\times dy\times du\times dj)\\ &\quad= \sum_{i\in\mathcal{S},y\in\mathcal{Y}} \pi_i \nu_y(i)\mathrm{b}(i,y)\nabla\left(\sum_{u\in\mathcal{U},j\in\mathcal{S}} \mu_u(y)\mathrm{p}_{ij}(u)\right) = 0.\end{aligned}$$

Thus an unbiased estimate of the integral

$$\int_{(i,y,u,j)\in\mathcal{S}\times\mathcal{Y}\times\mathcal{U}\times\mathcal{S}} \left(\pi_i \nu_y(i)\nabla\mu_u(y)\mathrm{p}_{ij}(u)\mathrm{J}_\beta(j) - \varphi(i,y,u,j)\right)\mathfrak{C}(di\times dy\times du\times dj)$$

is an unbiased estimate of $\nabla_\beta \eta$. Here results analogous to those achieved for $\varphi_{\mathcal{S}}(i,j)$ can be obtained. However, we focus on the more interesting (and practical) case of the restricted control variate

$$\varphi_{\mathcal{Y}}(i,y,u,j) \stackrel{\text{def}}{=} \pi_i \nu_y(i)\nabla\mu_u(y)\mathrm{p}_{ij}(u)\mathrm{b}_{\mathcal{Y}}(y).$$

Here, only information that can be observed by the controller (the observations y) may be used to minimize the variance. Recall, from Equation (9), we use $\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}})$ to denote the variance with such a restricted baseline control variate,

$$\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \mathrm{Var}_\pi\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\left(\mathrm{J}_\beta(j) - \mathrm{b}_{\mathcal{Y}}(y)\right)\right).$$

We use $\sigma^2_{\mathcal{Y}}$ to denote the variance without a baseline, that is

$$\sigma^2_{\mathcal{Y}} = \mathrm{Var}_\pi\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\mathrm{J}_\beta(j)\right).$$

We have the following theorem.

Theorem 13. *Let $D = (\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,\mathrm{r},\mu)$ be a controlled POMDP satisfying Assumptions 1, 2 and 3, with stationary distribution π. Then*

$$\sigma^2_{\mathcal{Y}}(\mathrm{b}^*_{\mathcal{Y}}) \stackrel{\text{def}}{=} \inf_{\mathrm{b}_{\mathcal{Y}}\in\mathbb{R}^{\mathcal{Y}}} \sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \sigma^2_{\mathcal{Y}} - \mathbb{E}_\pi\left[\frac{\left(\mathbb{E}_\pi\left[\left(\nabla\mu_u(y)/\mu_u(y)\right)^2\mathrm{J}_\beta(j)\middle|y\right]\right)^2}{\mathbb{E}_\pi\left[\left(\nabla\mu_u(y)/\mu_u(y)\right)^2\middle|y\right]}\right],$$

where $\mathbb{E}_\pi[\cdot|y]$ is the expectation (of π-distributed random variables, that is, random variables distributed as in $\mathbb{E}_\pi[\cdot]$) conditioned on observing y, and this infimum is attained with the baseline

$$\mathrm{b}^*_{\mathcal{Y}}(y) = \frac{\mathbb{E}_\pi\left[(\nabla\mu_u(y)/\mu_u(y))^2 \mathrm{J}_\beta(j)\middle| y\right]}{\mathbb{E}_\pi\left[(\nabla\mu_u(y)/\mu_u(y))^2\middle| y\right]}.$$

Furthermore, when restricted to the class of constant baselines, $b \in \mathbb{R}$, the minimal variance occurs with

$$b^* = \frac{\mathbb{E}_\pi\left[(\nabla\mu_u(y)/\mu_u(y))^2 \mathrm{J}_\beta(j)\right]}{\mathbb{E}_\pi(\nabla\mu_u(y)/\mu_u(y))^2}.$$

We have again used b^* to denote the optimal constant baseline. Note though that the b^* here differs from that given in Theorem 11. The proof uses the following lemma.

Lemma 14. *For any* $\mathrm{b}_{\mathcal{Y}}$*,*

$$\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \sigma^2_{\mathcal{Y}} + \mathbb{E}_\pi\left[\mathrm{b}^2_{\mathcal{Y}}(y)\,\mathbb{E}_\pi\left[\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2\middle| y\right] - 2\mathrm{b}_{\mathcal{Y}}(y)\mathbb{E}_\pi\left[\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2 J_\beta(j)\middle| y\right]\right].$$

Proof. Following the same steps as in the proof of Lemma 9,

$$\begin{aligned}
\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) &= \mathbb{E}_\pi\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\left(\mathrm{J}_\beta(j) - \mathrm{b}_{\mathcal{Y}}(y)\right) - \mathbb{E}_\pi\left[\frac{\nabla\mu_u(y)}{\mu_u(y)}\left(\mathrm{J}_\beta(j) - \mathrm{b}_{\mathcal{Y}}(y)\right)\right]\right)^2 \\
&= \sigma^2_{\mathcal{Y}} + \mathbb{E}_\pi\left[\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\mathrm{b}_{\mathcal{Y}}(y)\right)^2 - 2\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\mathrm{b}_{\mathcal{Y}}(y)\right)'\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\mathrm{J}_\beta(j)\right)\right] \\
&= \sigma^2_{\mathcal{Y}} + \sum_y\left[\mathrm{b}^2_{\mathcal{Y}}(y)\left(\sum_{i,u,j}\pi_i\nu_y(i)\mu_u(y)\mathrm{p}_{ij}(u)\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2\right)\right. \\
&\qquad \left. -2\mathrm{b}_{\mathcal{Y}}(y)\left(\sum_{i,u,j}\pi_i\nu_y(i)\mu_u(y)\mathrm{p}_{ij}(u)\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2\mathrm{J}_\beta(j)\right)\right].
\end{aligned}$$

Note that for functions $\mathrm{a}:\mathcal{Y}\to\mathbb{R}$ and $\mathrm{f}:\mathcal{S}\times\mathcal{Y}\times\mathcal{U}\times\mathcal{S}\to\mathbb{R}$

$$\begin{aligned}
&\sum_y \mathrm{a}(y)\sum_{\tilde{\imath},\tilde{u},\tilde{\jmath}}\pi_{\tilde{\imath}}\nu_y(\tilde{\imath})\mu_{\tilde{u}}(y)\mathrm{p}_{\tilde{\imath}\tilde{\jmath}}(\tilde{u})\mathrm{f}(\tilde{\imath},y,\tilde{u},\tilde{\jmath}) \\
&\quad= \sum_y \mathrm{a}(y)\sum_i \pi_i\nu_y(i)\sum_{\tilde{\imath},\tilde{y},\tilde{u},\tilde{\jmath}}\frac{\delta_{y\tilde{y}}\pi_{\tilde{\imath}}\nu_y(\tilde{\imath})\mu_{\tilde{u}}(y)\mathrm{p}_{\tilde{\imath}\tilde{\jmath}}(\tilde{u})}{\sum_i\pi_i\nu_y(i)}\mathrm{f}(\tilde{\imath},\tilde{y},\tilde{u},\tilde{\jmath}) \\
&\quad= \sum_{i,y}\pi_i\nu_y(i)\mathrm{a}(y)\sum_{\tilde{\imath},\tilde{y},\tilde{u},\tilde{\jmath}}\Pr\{\tilde{\imath},\tilde{y},\tilde{u},\tilde{\jmath}|\tilde{y}=y\}\,\mathrm{f}(\tilde{\imath},\tilde{y},\tilde{u},\tilde{\jmath}) \\
&\quad= \mathbb{E}_\pi[\mathrm{a}(y)\mathbb{E}_\pi[\mathrm{f}(i,y,u,j)|y]],
\end{aligned}$$

implying the result. ∎

Proof of Theorem 13. We apply Lemma 14 and minimize for each $\mathrm{b}_{\mathcal{Y}}(y)$ independently, to obtain

$$\mathrm{b}_{\mathcal{Y}}(y) = \frac{\mathbb{E}_\pi\left[\left(\nabla\mu_u(y)/\mu_u(y)\right)^2 \mathrm{J}_\beta(j)\middle| y\right]}{\mathbb{E}_\pi\left[\left(\nabla\mu_u(y)/\mu_u(y)\right)^2\middle| y\right]}.$$

Substituting gives the optimal variance. A similar argument gives the optimal constant baseline. ■

Example 1. *Consider the k-armed bandit problem (for example, see Sutton and Barto, 1998). Here each action is taken independently and the resultant state depends only on the action performed; that is $\mu_u(y) = \mu_u$ and $\mathrm{p}_{ij}(u) = \mathrm{p}_j(u)$. So, writing $R_\beta = \mathbb{E}_{U_0\sim\mu}\left[\sum_{t=1}^\infty \beta^t \mathrm{r}(X_t)\right]$, we have*

$$\begin{aligned}
\nabla_\beta\eta &= \mathbb{E}_\pi\left[\frac{\nabla\mu_u(y)}{\mu_u(y)}\mathrm{J}_\beta(j)\right] \\
&= \mathbb{E}_{u\sim\mu}\left[\frac{\nabla\mu_u}{\mu_u}\left(\mathrm{r}(j)+R_\beta\right)\right] \\
&= \mathbb{E}_{u\sim\mu}\left[\frac{\nabla\mu_u}{\mu_u}\mathrm{r}(j)\right].
\end{aligned}$$

Note that this last line is β independent, and it follows from $\lim_{\beta\to 1}\nabla_\beta\eta = \nabla\eta$ that

$$\nabla\eta = \nabla_\beta\eta \qquad \forall\beta\in[0,1]. \tag{15}$$

For $k=2$ (2 actions $\{u_1,u_2\}$) we have $\mu_{u_1}+\mu_{u_2}=1$ and $\nabla\mu_{u_1} = -\nabla\mu_{u_2}$, and so the optimal constant baseline is given by

$$\begin{aligned}
b^* &= \frac{\mathbb{E}_\pi\left[\left(\nabla\mu_u(y)/\mu_u(y)\right)^2\mathrm{J}_\beta(j)\right]}{\mathbb{E}_\pi\left(\nabla\mu_u(y)/\mu_u(y)\right)^2} \\
&= \frac{\mathbb{E}_{u\sim\mu}\left[\left(\nabla\mu_u/\mu_u\right)^2\mathrm{r}(j)\right]}{\mathbb{E}_{u\sim\mu}\left(\nabla\mu_u/\mu_u\right)^2} + R_\beta \\
&= \frac{\mu_{u_1}\left(\nabla\mu_{u_1}/\mu_{u_1}\right)^2\mathbb{E}\left[\mathrm{r}|u_1\right] + \mu_{u_2}\left(\nabla\mu_{u_2}/\mu_{u_2}\right)^2\mathbb{E}\left[\mathrm{r}|u_2\right]}{\mu_{u_1}\left(\nabla\mu_{u_1}/\mu_{u_1}\right)^2 + \mu_{u_2}\left(\nabla\mu_{u_2}/\mu_{u_2}\right)^2} + R_\beta \\
&= \frac{\mu_{u_1}\mu_{u_2}}{\mu_{u_1}+\mu_{u_2}}\left(\frac{1}{\mu_{u_1}}\mathbb{E}\left[\mathrm{r}|u_1\right] + \frac{1}{\mu_{u_2}}\mathbb{E}\left[\mathrm{r}|u_2\right]\right) + R_\beta \\
&= \mu_{u_2}\mathbb{E}\left[\mathrm{r}|u_1\right] + \mu_{u_1}\mathbb{E}\left[\mathrm{r}|u_2\right] + R_\beta,
\end{aligned}$$

where we have used $\mathbb{E}\left[\mathrm{r}|u\right]$ to denote $\mathbb{E}_{j\sim\mathrm{p}(u)}\mathrm{r}(j)$. From (15) we know that β may be chosen arbitrarily. Choosing $\beta=0$ gives $R_\beta = 0$ and we regain the result of Dayan (1990).

In the special case of a controlled MDP we obtain the result that would be expected. This follows immediately from Theorem 13.

Corollary 15. *Let $D = (\mathcal{S},\mathcal{U},P,\mathrm{r},\mu)$ be a controlled MDP satisfying Assumptions 1, 2 and 3, with stationary distribution π. Then*

$$\inf_{\mathrm{b}_{\mathcal{Y}}\in\mathbb{R}^{\mathcal{S}}}\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \sigma^2_{\mathcal{Y}} - \mathbb{E}_{i\sim\pi}\left[\frac{\left(\mathbb{E}\left[\left(\nabla\mu_u(i)/\mu_u(i)\right)^2\mathrm{J}_\beta(j)\middle| i\right]\right)^2}{\mathbb{E}\left[\left(\nabla\mu_u(i)/\mu_u(i)\right)^2\middle| i\right]}\right],$$

and this infimum is attained with the baseline

$$\mathrm{b}_{\mathcal{Y}}(i) = \frac{\mathbb{E}\left[\left(\nabla\mu_u(i)/\mu_u(i)\right)^2 \mathrm{J}_\beta(j)\middle| i\right]}{\mathbb{E}\left[\left(\nabla\mu_u(i)/\mu_u(i)\right)^2\middle| i\right]}.$$

The following theorem shows that, just as in the Markov chain case, the variance of an estimate with an arbitrary baseline can be expressed as the sum of the variance with the optimal baseline and a certain squared weighted distance between the baseline function and the optimal baseline function.

Theorem 16. *Let $(\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,\mathrm{r},\mu)$ be a controlled POMDP satisfying Assumptions 1, 2 and 3, with stationary distribution π. Then*

$$\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) - \sigma^2_{\mathcal{Y}}(\mathrm{b}^*_{\mathcal{Y}}) = \mathbb{E}_\pi\left[\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2\left(\mathrm{b}_{\mathcal{Y}}(y) - \mathrm{b}^*_{\mathcal{Y}}(y)\right)^2\right].$$

Furthermore if the estimate using b^*, the optimal constant *baseline defined in Theorem 13, has variance $\sigma^2_{\mathcal{Y}}(b^*)$, we have that the variance $\sigma^2_{\mathcal{Y}}(b)$ of the gradient estimate with an arbitrary constant baseline is*

$$\sigma^2_{\mathcal{Y}}(b) - \sigma^2_{\mathcal{Y}}(b^*) = \mathbb{E}_\pi\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2 (b-b^*)^2.$$

Proof. For each $y \in \mathcal{Y}$, define S_y and W_y as

$$\begin{aligned} S_y &= \mathbb{E}\left[\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2\middle| y\right], \\ W_y &= \mathbb{E}\left[\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\right)^2 J_\beta(j)\middle| y\right]. \end{aligned}$$

Follow the steps in Theorem 10, replacing S_i with S_y, and W_i with W_y. The constant baseline case follows similarly by considering $S = \mathbb{E}_\pi S_y$ and $W = \mathbb{E}_\pi W_y$. ■

In Section 7.1 we will see how Theorem 16 can be used to construct a practical algorithm for finding a good baseline. In most cases it is not possible to calculate the optimal baseline, $\mathrm{b}^*_{\mathcal{Y}}$, a priori. However, for a parameterized class of baseline functions, a gradient descent approach could be used to find a good baseline. Section 7.1 explores this idea.

As before, Theorem 16 also gives us a tool to measure how far the baseline $b = \mathbb{E}_\pi \mathrm{J}_\beta(i)$ is from the optimum.

Corollary 17. *Let $D = (\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,\mathrm{r},\mu)$ be a controlled POMDP satisfying Assumptions 1, 2 and 3, with stationary distribution π. Then*

$$\sigma^2_{\mathcal{Y}}(\mathbb{E}_\pi \mathrm{J}_\beta(i)) - \inf_{b\in\mathbb{R}} \sigma^2_{\mathcal{Y}}(b) = \frac{\left(\mathbb{E}_\pi\left(\nabla\mu_u(y)/\mu_u(y)\right)^2 \mathbb{E}_\pi \mathrm{J}_\beta(j) - \mathbb{E}_\pi\left[\left(\nabla\mu_u(y)/\mu_u(y)\right)^2 \mathrm{J}_\beta(j)\right]\right)^2}{\mathbb{E}_\pi\left(\nabla\mu_u(y)/\mu_u(y)\right)^2}.$$

As in the case of a Markov reward process, the sub-optimality of the choice $b = \mathbb{E}_\pi \mathrm{J}_\beta(i)$ depends on the independence of the random variables $(\nabla\mu_u(y)/\mu_u(y))^2$ and $\mathrm{J}_\beta(j)$; if they are nearly independent, $\mathbb{E}_\pi \mathrm{J}_\beta(i)$ is a good choice.

6. Value Functions: Actor-Critic Methods

Consider the estimate produced by GPOMDP (see Equation (5)) in the MDP setting, where the state is observed. In this section we look at replacing J_t, the biased and noisy estimate of the discounted value function, in Δ_T with an arbitrary value function, that is, a function $\mathrm{V} : \mathcal{S} \to \mathbb{R}$. For a MDP, this gives the following estimate of $\nabla_\beta \eta$:

$$\Delta_T^{\mathrm{V}} \stackrel{\text{def}}{=} \frac{1}{T} \sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(X_t)}{\mu_{U_t}(X_t)} \mathrm{V}(X_{t+1}). \tag{16}$$

Imagine that the discounted value function, J_β, is known. By replacing J_t with $\mathrm{J}_\beta(X_t)$ in Equation (5), that is, by choosing $\mathrm{V} = \mathrm{J}_\beta$, the bias and noise due to J_t is removed. This seems a good choice, but we may be able to do better. Indeed we will see that in some cases the selection of a value function differing from the discounted value function can remove *all* estimation variance, whilst introducing no bias.

6.1 Control Variate for a Value Function

Consider a control variate of the form

$$\varphi_\beta(i,u,j) \stackrel{\text{def}}{=} \pi_i \nabla \mu_u(i) \mathrm{p}_{ij}(u) \mathrm{A}_\beta(j)$$

where

$$\mathrm{A}_\beta(j) \stackrel{\text{def}}{=} \lim_{T\to\infty} \mathbb{E}\left[\sum_{k=1}^{T} \beta^{k-1} \mathrm{d}(X_{t+k}, X_{t+1+k}) \middle| X_{t+1} = j \right]$$

and

$$\mathrm{d}(i,j) \stackrel{\text{def}}{=} \mathrm{r}(i) + \beta \mathrm{V}(j) - \mathrm{V}(i).$$

We make the following assumption.

Assumption 5. *For all* $j \in \mathcal{S}$, $|\mathrm{V}(j)| \leq \mathbf{M} < \infty$.

Under this assumption, the estimation of the integral

$$\int_{(i,u,j)\in\mathcal{S}\times\mathcal{U}\times\mathcal{S}} \left(\pi_i \nabla \mu_u(i) \mathrm{p}_{ij}(u) \mathrm{J}_\beta(j) - \varphi_\beta(i,u,j) \right) \mathfrak{C}(di \times du \times dj) \tag{17}$$

has an expected bias from $\nabla_\beta \eta$ of

$$\int_{(i,u,j)\in\mathcal{S}\times\mathcal{U}\times\mathcal{S}} \varphi_\beta(i,u,j) \mathfrak{C}(di \times du \times dj) = \sum_{i\in\mathcal{S}, u\in\mathcal{U}, j\in\mathcal{S}} \pi_i \nabla \mu_u(i) \mathrm{p}_{ij}(u) \left(\mathrm{J}_\beta(j) - \mathrm{V}(j) \right).$$

This can be easily seen by noting that under Assumption 5, and as $\beta \in [0,1)$,

$$\begin{aligned} \mathrm{A}_\beta(j) &= \lim_{T\to\infty} \mathbb{E}\left[\sum_{k=1}^{T} \beta^{k-1} \left(\mathrm{r}(X_{t+k}) + \beta \mathrm{V}(X_{t+1+k}) - \mathrm{V}(X_{t+k}) \right) \middle| X_{t+1} = j \right] \\ &= \mathrm{J}_\beta(j) - \mathrm{V}(j) + \lim_{T\to\infty} \mathbb{E}\left[\beta^T \mathrm{V}(X_{t+1+T}) \middle| X_{t+1} = j \right] \\ &= \mathrm{J}_\beta(j) - \mathrm{V}(j). \end{aligned}$$

We see then that Δ_T^{V} gives an estimate of the integral in Equation (17). The following theorem gives a bound on the expected value of the squared Euclidean distance between this estimate and $\nabla_\beta \eta$. Notice that the bound includes both bias and variance terms.

Theorem 18. *Let $D = (\mathcal{S}, \mathcal{U}, P, \mathrm{r}, \mu)$ be a controlled MDP satisfying Assumptions 1, 2 and 3, with stationary distribution π. Let $\{X_t, U_t\}$ be a process generated by D, starting $X_0 \sim \pi$. Then*

$$\mathbb{E}\left(\Delta_T^{\mathrm{V}} - \nabla_\beta \eta\right)^2 = \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(X_t)}{\mu_{U_t}(X_t)} \mathrm{V}(X_{t+1})\right) + \left(\mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{A}_\beta(j)\right]\right)^2,$$

and hence there is an Ω^* such that

$$\mathbb{E}\left(\Delta_T^{\mathrm{V}} - \nabla_\beta \eta\right)^2 \leq \frac{\Omega^*}{T} \mathrm{Var}_\pi\left(\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j)\right) + \left(\mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{A}_\beta(j)\right]\right)^2.$$

Proof.

$$\begin{aligned}
&\mathbb{E}\left(\Delta_T^{\mathrm{V}} - \nabla_\beta \eta\right)^2 \\
&= \mathbb{E}\left(\frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(X_t)}{\mu_{U_t}(X_t)} \mathrm{V}(X_{t+1}) - \mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)}\left(\mathrm{V}(j) + \mathrm{A}_\beta(j)\right)\right]\right)^2 \\
&= \mathbb{E}\left(\frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(X_t)}{\mu_{U_t}(X_t)} \mathrm{V}(X_{t+1}) - \mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j)\right]\right)^2 \\
&\quad - 2\left(\mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{A}_\beta(j)\right]\right)' \left(\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(X_t)}{\mu_{U_t}(X_t)} \mathrm{V}(X_{t+1}) - \mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j)\right]\right]\right) \\
&\quad + \left(\mathbb{E}\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{A}_\beta(j)\right]\right)^2 \qquad (18) \\
&= \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(X_t)}{\mu_{U_t}(X_t)} \mathrm{V}(X_{t+1})\right) + \left(\mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{A}_\beta(j)\right]\right)^2 \\
&\leq \frac{\Omega^*}{T} \mathrm{Var}_\pi\left(\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j)\right) + \left(\mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{A}_\beta(j)\right]\right)^2. \qquad (19)
\end{aligned}$$

Note that

$$\mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j)\right] = \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla \mu_{U_t}(X_t)}{\mu_{U_t}(X_t)} \mathrm{V}(X_{t+1})\right],$$

which means that the second term of Equation (18) is zero, and the first term becomes the variance of the estimate. Equation (19), and hence Theorem 18, follow from Theorem 3. ■

Corollary 19. *Let $D = (\mathcal{S}, \mathcal{U}, P, \mathrm{r}, \mu)$ be a controlled MDP satisfying Assumptions 1, 2 and 3. Let $M = (\mathcal{S}, P)$ be the resultant chain of states, and let π be its stationary distribution; M has mixing*

time τ. *Let* $\{X_t, U_t\}$ *be a process generated by D, starting* $X_0 \sim \pi$. *Then for any* $0 < \varepsilon < e^{-1}$ *there is a* $C_\varepsilon \leq 1 + 50\tau(1+\mathbf{M}) + 8\tau \ln \varepsilon^{-1}$ *such that*

$$\mathbb{E}\left(\Delta_T^{\mathrm{V}} - \nabla_\beta \eta\right)^2 \leq K\varepsilon + \frac{C_\varepsilon}{T}\mathrm{Var}_\pi\left(\frac{\nabla\mu_u(i)}{\mu_u(i)}\mathrm{V}(j)\right) + \left(\mathbb{E}_\pi\left[\frac{\nabla\mu_u(i)}{\mu_u(i)}\mathrm{A}_\beta(j)\right]\right)^2.$$

Proof. Apply Theorem 4 to the first part of Theorem 18, for each of the K dimensions, noting that the mixing time of the process $\{X_t, U_t, X_{t+1}\}$ is at most $\tau\ln(2e) \leq 2\tau$ (Lemma 1). ■

6.2 Zero Variance, Zero Bias Example

Write $v = \mathrm{V} - \mathrm{J}_\beta$. The bias due to using V in place of J_β is given by

$$Gv,$$

where G is a $K \times |\mathcal{S}|$ matrix with its j^{th} column given by $\sum_{i\in\mathcal{S}, u\in\mathcal{U}} \pi_i \nabla\mu_u(i)\mathrm{p}_{ij}(u)$. If v is in the right null space of G then this bias is zero. An example of such a v is a constant vector; $v = (c, c, \ldots, c)'$. This can be used to construct a trivial example of how Δ_T^{V} (Equation (16)) can produce an unbiased, zero variance estimate. The observation that we need only consider value functions that span the range space of G to produce a "good" gradient estimate, in the sense that convergence results may be obtained, was made by Konda and Tsitsiklis (2003, 2000); Sutton et al. (2000). Here we wish to consider a richer class of value functions for the purpose of actively reducing the variance of gradient estimates.

Consider a controlled MDP $D = (\mathcal{S}, \mathcal{U}, P, \mathrm{r}, \mu)$ satisfying Assumptions 1, 2 and 3, and with $\mathrm{r}(i) = (1-\beta)c$, for some constant c, and all $i \in \mathcal{S}$. This gives a value function of $\mathrm{J}_\beta(i) = c$, for all $i \in \mathcal{S}$, and consequently

$$\nabla_\beta \eta = \sum_{i,u} \pi_i \nabla\mu_u(i) c = c \sum_i \pi_i \nabla \sum_u \mu_u(i) = 0.$$

With $v = (-c, -c, \ldots, -c)'$, and selecting the fixed value function $\mathrm{V} = \mathrm{J}_\beta + v$, we have

$$\frac{\nabla\mu_u(i)}{\mu_u(i)}\mathrm{V}(j) = 0, \qquad \forall i, u, j.$$

So Δ_T^{V} will produce a zero bias, zero variance estimate of $\nabla_\beta\eta$. Note also that if the MDP is such that there exists an i,u pair such that $\Pr\{X_t = i, U_t = u\} > 0$ and $\nabla\mu_u(i) \neq 0$ then selecting $\mathrm{V} = \mathrm{J}_\beta$ gives an estimate that, whilst still unbiased, has non-zero variance. The event

$$\left\{\frac{\nabla\mu_u(i)}{\mu_u(i)}\mathrm{V}(j) \neq 0\right\}$$

has a non-zero probability of occurrence.

A less trivial example is given in Appendix D.

7. Algorithms

In Section 5 and Section 6 we have seen bounds on squared error of gradient estimates when using various additive control variates. For the baseline control variates we have also seen the choice of baseline which minimizes this bound. Though it may not be possible to select the best baseline or value function a priori, data could be used to help us choose. For a parameterized baseline, or value function, we could improve the error bounds via gradient decent. In this section we explore this idea.

7.1 Minimizing Weighted Squared Distance to the Optimal Baseline

Given a controlled POMDP and a parameterized class of baseline functions

$$\left\{ \mathrm{b}_{\mathcal{Y}}(\cdot,\omega) : \mathcal{Y} \to \mathbb{R} \,\middle|\, \omega \in \mathbb{R}^L \right\},$$

we wish to choose a baseline function to minimize the variance of our gradient estimates. Theorem 16 expresses this variance as the sum of the optimal variance and a squared distance between the baseline function and the optimal one. It follows that we can minimize the variance of our gradient estimates by minimizing the distance between our baseline and the optimum baseline. The next theorem shows that we can use a sample path of the controlled POMDP to estimate the gradient (with respect to the parameters of the baseline function) of this distance. We need to make the following assumptions about the parameterized baseline functions.

Assumption 6. *There are bounds $\mathbf{M}, \mathbf{G} < \infty$ such that for all $y \in \mathcal{Y}$, and all $\omega \in \mathbb{R}^L$, the baseline function is bounded, $\left|\mathrm{b}_{\mathcal{Y}}(y,\omega)\right| \leq \mathbf{M}$, and the gradient of the baseline is bounded, $\left\|\nabla \mathrm{b}_{\mathcal{Y}}(y,\omega)\right\| \leq \mathbf{G}$.*

We drop ω in the notation, and, to avoid confusion, we write $\mathrm{g}^2(y,u)$ to denote $[(\nabla \mu_u(y))/\mu_u(y)]^2$, where the gradient is with respect to the parameters of the policy, θ.

Theorem 20. *Let $D = (\mathcal{S}, \mathcal{U}, \mathcal{Y}, P, \nu, \mathrm{r}, \mu)$ be a controlled POMDP satisfying Assumptions 1, 2 and 3. Let $\mathrm{b}_{\mathcal{Y}} : \mathcal{Y} \times \mathbb{R}^L \to \mathbb{R}$ be a parameterized baseline function satisfying Assumption 6. If $\{X_t, Y_t, U_t\}$ is a sample path of the controlled POMDP (for any X_0), then with probability* 1

$$\frac{1}{2}\nabla \sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left(\mathrm{b}_{\mathcal{Y}}(Y_{t-1}) - \beta \mathrm{b}_{\mathcal{Y}}(Y_t) - \mathrm{r}(X_t)\right) \sum_{s=0}^{t-1} \beta^{t-s-1} \nabla \mathrm{b}_{\mathcal{Y}}(Y_s) \mathrm{g}^2(Y_s, U_s).$$

Proof. From Theorem 16,

$$\frac{1}{2}\nabla \sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \mathbb{E}_\pi \left[\mathrm{g}^2(y,u) \nabla \mathrm{b}_{\mathcal{Y}}(y) \left(\mathrm{b}_{\mathcal{Y}}(y) - \mathrm{b}^*_{\mathcal{Y}}(y) \right) \right],$$

but

$$\begin{aligned}
&\mathbb{E}_\pi \left[\mathrm{g}^2(y,u) \nabla \mathrm{b}_{\mathcal{Y}}(y) \mathrm{b}^*_{\mathcal{Y}}(y) \right] \\
&\quad = \sum_{i,y,u} \pi_i \nu_y(i) \mu_u(y) \mathrm{g}^2(y,u) \nabla \mathrm{b}_{\mathcal{Y}}(y) \\
&\qquad\qquad \times \frac{\sum_{\tilde{\imath},\tilde{u},\tilde{\jmath}} \pi_{\tilde{\imath}} \nu_y(\tilde{\imath}) \mu_{\tilde{u}}(y) \mathrm{p}_{\tilde{\jmath}}(\tilde{u}) \mathrm{g}^2(y,\tilde{u}) \mathrm{J}_\beta(\tilde{\jmath})}{\sum_{\tilde{\imath},\tilde{u}} \pi_{\tilde{\imath}} \nu_y(\tilde{\imath}) \mu_{\tilde{u}}(y) \mathrm{g}^2(y,\tilde{u})} \\
&\quad = \sum_y \nabla \mathrm{b}_{\mathcal{Y}}(y) \sum_{i,u,j} \pi_i \nu_y(i) \mu_u(y) \mathrm{p}_{ij}(u) \mathrm{g}^2(y,u) \mathrm{J}_\beta(j) \\
&\quad = \mathbb{E}_\pi \left[\nabla \mathrm{b}_{\mathcal{Y}}(y) \mathrm{g}^2(y,u) \mathrm{J}_\beta(j) \right].
\end{aligned}$$

Also, as $b_\gamma(Y_t)$ is uniformly bounded, we can write

$$b_\gamma(Y_t) = \sum_{s=t+1}^{\infty} \beta^{s-t-1}(b_\gamma(Y_{s-1}) - \beta b_\gamma(Y_s)).$$

The boundedness of r, and the dominated convergence theorem, likewise gives $J_\beta(X_{t+1}) = \mathbb{E}[\sum_{s=t+1}^{\infty} \beta^{s-t-1} r(X_s)|X_{t+1}]$. Now we have

$$\frac{1}{2}\nabla\sigma^2_\gamma(b_\gamma) = \mathbb{E}_\pi\left[g^2(Y_t,U_t)\nabla b_\gamma(Y_t)\sum_{s=t+1}^{\infty}\beta^{s-t-1}\left(b_\gamma(Y_{s-1}) - \beta b_\gamma(Y_s) - r(X_s)\right)\right]. \tag{20}$$

The rest of the proof is as the proof of Baxter and Bartlett (2001, Theorem 4): we use an ergodicity result to express the expectation as an average, then show that we can truncate the tail of the β decaying sum.

Assume $X_0 \sim \pi$. Write $\tilde{X}_t$ to denote the tuple (X_t, Y_t, U_t), write $\tilde{P}$ to denote the corresponding transition matrix, and write $\tilde{\pi}$ to denote the corresponding stationary distribution (so $\tilde{\pi}_{i,y,u} = \pi_i \nu_y(i)\mu_u(y)$). Now consider running the Markov chain on the process $\{\tilde{X}_t\}$ backwards. We have

$$\Pr\left\{\tilde{X}_{-1}|\tilde{X}_0,\tilde{X}_1,\ldots\right\} = \frac{\Pr\left\{\tilde{X}_{-1},\tilde{X}_0,\tilde{X}_1,\ldots\right\}}{\Pr\left\{\tilde{X}_0,\tilde{X}_1,\ldots\right\}} = \frac{\Pr\left\{\tilde{X}_{-1}\right\}\tilde{P}_{\tilde{X}_{-1}\tilde{X}_0}}{\tilde{\pi}_{\tilde{X}_0}} = \frac{\tilde{\pi}_{\tilde{X}_{-1}}\tilde{P}_{\tilde{X}_{-1}\tilde{X}_0}}{\tilde{\pi}_{\tilde{X}_0}},$$

as $\tilde{\pi}$ is the unique distribution such that $\tilde{\pi}'\tilde{P} = \tilde{\pi}'$. This gives the distribution for $\tilde{X}_{-1}$, and repeating this argument gives the distribution for $\tilde{X}_{-2}, \tilde{X}_{-3}, \ldots$. Denote this doubly infinite process by $\{\tilde{X}_t\}_{-\infty}^{\infty}$. We wish to look at the behavior of time averages of the function

$$f\left(\left\{\tilde{X}_t\right\}_{-\infty}^{\infty}\right) \stackrel{\text{def}}{=} g^2(Y_0,U_0)\nabla b_\gamma(Y_0)\sum_{s=1}^{\infty}\beta^{s-1}\left(b_\gamma(Y_{s-1}) - \beta b_\gamma(Y_s) - r(X_s)\right).$$

Specifically, we would like to show that

$$\lim_{T\to\infty}\frac{1}{T}\sum_{m=0}^{T-1} f\left(\mathfrak{S}^m\left(\left\{\tilde{X}_t\right\}_{-\infty}^{\infty}\right)\right) = \mathbb{E}\left[f\left(\left\{\tilde{X}_t\right\}_{-\infty}^{\infty}\right)\right], \qquad \text{w.p.1} \tag{21}$$

where $\mathfrak{S}^m$ denotes m applications of the shift operator $\mathfrak{S}$, and where $\mathfrak{S}(\{\tilde{X}_t\}_{-\infty}^{\infty}) = \{W_t\}_{-\infty}^{\infty}$ with $W_t = \tilde{X}_{t+1}$ for all t. Doob (1994, L^2 Ergodic theorem, pg. 119) tells us, provided that $\mathfrak{S}$ is one-to-one and measure preserving, and that f is square integrable, the left hand side of Equation (21) is almost surely constant, and furthermore, provided that the only invariant sets of $\mathfrak{S}$ are sets of measure zero and their complements, this constant is equal to the right hand side of Equation (21). Expanding f and $\mathfrak{S}$ in Equation (21) then gives, with probability one,

$$\frac{1}{2}\nabla\sigma^2_\gamma(b_\gamma) = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} g^2(Y_t,U_t)\nabla b_\gamma(Y_t)\sum_{s=t+1}^{\infty}\beta^{s-t-1}\left(b_\gamma(Y_{s-1}) - \beta b_\gamma(Y_s) - r(X_s)\right). \tag{22}$$

It remains to be shown that the conditions of the L^2 Ergodic theorem hold.

1. $\mathfrak{S}$ *is one-to-one.* By considering how $\mathfrak{S}$ behaves at each index, we see that it is a bijection.

2. $\mathfrak{S}$ *is measure preserving.* That is, for a set of sequences A, A and $\mathfrak{S}(A)$ have the same measure. This follows from the Markov property, and from the fact that the transition operator at time t, as well as the marginal distribution on X_t, is identical for all times t. Specifically, we have the following. Let $A^- \subset A$ be the smallest set such that $\Pr\{\{W_t\}_{-\infty}^{\infty} \in A\} = \Pr\{\{W_t\}_{-\infty}^{\infty} \in A^- \cap A\}$, and write $A_s = \{W_s : \{W_t\}_{-\infty}^{\infty} \in A^-\}$ and $A_{s_1}^{s_2} = \{\{W_t\}_{s_1}^{s_2} : \{W_t\}_{-\infty}^{\infty} \in A^-\}$, where $\{W_t\}_{s_1}^{s_2}$ is the process starting at $t = s_1$ and ending at $t = s_2$. For any s we have

$$\begin{aligned}
\Pr\left\{\mathfrak{S}\left(\{\tilde{X}_t\}_{-\infty}^{\infty}\right) \in A\right\} &= \int_{x \in A_s} \Pr\{\tilde{X}_{s+1} = x\} \Pr\left\{\{\tilde{X}_t\}_{s+2}^{\infty} \in A_{s+1}^{\infty} \middle| \tilde{X}_{s+1} = x\right\} \\
&\qquad \times \Pr\left\{\{\tilde{X}_t\}_{-\infty}^{s} \in A_{-\infty}^{s-1} \middle| \tilde{X}_{s+1} = x\right\} \mathfrak{C}(dx) \\
&= \int_{x \in A_s} \Pr\{\tilde{X}_s = x\} \Pr\left\{\{\tilde{X}_t\}_{s+1}^{\infty} \in A_{s+1}^{\infty} \middle| \tilde{X}_s = x\right\} \\
&\qquad \times \Pr\left\{\{\tilde{X}_t\}_{-\infty}^{s-1} \in A_{-\infty}^{s-1} \middle| \tilde{X}_s = x\right\} \mathfrak{C}(dx) \\
&= \Pr\left\{\{\tilde{X}_t\}_{-\infty}^{\infty} \in A\right\}.
\end{aligned}$$

   We also have that $\mathfrak{S}^{-1}$ (the inverse of $\mathfrak{S}$) is measure preserving; by the change of variables $\{W_t\}_{-\infty}^{\infty} = \mathfrak{S}(\{\tilde{X}_t\}_{-\infty}^{\infty})$.

3. f *is square integrable.* The measure on $\{\tilde{X}_t\}_{-\infty}^{\infty}$ is finite, and $|\mathrm{f}|$ is bounded.

4. *If set A is such that $\mathfrak{S}^{-1}(A) = A$ (where $\mathfrak{S}^{-1}(A) = \{\{W_t\}_{-\infty}^{\infty} : \mathfrak{S}(\{W_t\}_{-\infty}^{\infty}) \in A\}$), then either A has measure zero, or $\bar{A}$ has measure one.* Consider a set A of positive measure such that $\mathfrak{S}^{-1}(A) = A$, and write $\bar{A}$ for its complement. As $\mathfrak{S}$ is a bijection, we also have that $\mathfrak{S}^{-1}(\bar{A}) = \bar{A}$. Assumption 1 implies that $A_t = \tilde{S}$ (at least, this is true for the state component, and, without loss of generality, we may assume it is true for the extended space). If we assume that $A_0 \cap \bar{A}_0 = \emptyset$, and hence $A_t \cap \bar{A}_t = \emptyset$ for all t, then the measure of $\bar{A}$ must be zero. We will show that $A_0 \cap \bar{A}_0 = \emptyset$.

   Unless $\bar{A}$ has measure zero, for each $x \in A_0 \cap \bar{A}_0$ we must have that $\Pr\{\{W_t\}_{-\infty}^{-1} \in \bar{A}_{-\infty}^{-1} | W_0 = x\}$ and $\Pr\{\{W_t\}_1^{\infty} \in A_1^{\infty} | W_0 = x\}$ are both positive, by the Markov property. Hence if $A_0 \cap \bar{A}_0$ is non-empty there must be a set of positive measure, which we denote B, that follows sequences in $\bar{A}^-$ until time $t = 0$, and then follows sequences in A^-. Without loss of generality, let us assume that $B \subset A^-$. We will also assume that if $b \in B$ then $\mathfrak{S}^{-1}(b) \in B$, as the existence of B implies the existence of $\hat{B} = B \cup \{\mathfrak{S}^{-1}(b)\}$ with the same properties. We will show that such a B does not exist, and therefore $A_0 \cap \bar{A}_0 = \emptyset$.

   Let $A_{-\infty}^{s*} = \{\{W_t\}_{-\infty}^{\infty} : \{W_t\}_{-\infty}^{s} \in A_{-\infty}^{s}\}$, the set of sequences that follow A^- until time s, and then follow any sequence. We have that $B_{-\infty}^{0*} \subset \bar{A}_{-\infty}^{0*}$. Construct the set $B^* \subset B$ by $B^* = \lim_{t\to\infty} \mathfrak{S}^{-t}(B)$ (note, $\mathfrak{S}^{-t}(B)$ is a non-increasing sequence of sets, and hence its limit exists). We have that $\mathfrak{S}^{-t}(B) \subset \mathfrak{S}^{-t}(B)_{-\infty}^{t*} = \mathfrak{S}^{-t}(B_{-\infty}^{0*}) \subset \mathfrak{S}^{-t}(\bar{A}_{-\infty}^{0*}) = \bar{A}_{-\infty}^{t*}$, and so $B^* = \liminf_{t\to\infty} \mathfrak{S}^{-t}(B) \subset \limsup_{t\to\infty} \bar{A}_{-\infty}^{t*} = \bar{A}^-$, where the last equality follows from $\bar{A}_{-\infty}^{t*}$ being non-increasing. Furthermore, by the dominated convergence theorem we have $\Pr\{\{W_s\}_{-\infty}^{\infty} \in B^*\} = \lim_{t\to\infty} \Pr\{\{W_s\}_{-\infty}^{\infty} \in \mathfrak{S}^{-t}(B)\} = \Pr\{\{W_s\}_{-\infty}^{\infty} \in B\} > 0$. This means that the set B^* has positive measure and is a subset of both A and $\bar{A}$, which is impossible, and so such a B does not exist.

The statement given by Equation (21) is for a sample such that $X_0 \sim \pi$, but can be generalized to an arbitrary distribution using the convergence of $\{X_t\}$ to stationarity. Indeed, for the finite chains we consider, all states have positive π-measure, and hence Equation (22) holds for $X_0 \sim \pi$ only if it holds for all $X_0 \in \mathcal{S}$.

If we truncate the inner sum at T, the norm of the error is

$$\begin{aligned}
&\left\| \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathrm{g}^2(Y_t,U_t)\nabla \mathrm{b}_{\mathcal{Y}}(Y_t) \sum_{s=T+1}^{\infty} \beta^{s-t-1}\left(\mathrm{b}_{\mathcal{Y}}(Y_{s-1}) - \beta \mathrm{b}_{\mathcal{Y}}(Y_s) - \mathrm{r}(X_s)\right)\right\| \\
&= \left\| \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathrm{g}^2(Y_t,U_t)\nabla \mathrm{b}_{\mathcal{Y}}(Y_t)\beta^T \left(\mathrm{b}_{\mathcal{Y}}(Y_T) - \sum_{s=T+1}^{\infty} \beta^{s-t-1}\mathrm{r}(X_s)\right)\right\| \\
&\leq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{B}^2\mathbf{G}\beta^T \left(\mathbf{M} + \frac{\mathbf{R}}{1-\beta}\right) \\
&= 0,
\end{aligned}$$

where we have used Assumptions 2, 3, and 6. This gives

$$\frac{1}{2}\nabla \sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}}) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathrm{g}^2(Y_t,U_t)\nabla \mathrm{b}_{\mathcal{Y}}(Y_t) \sum_{s=t+1}^{T} \beta^{s-t-1}\left(\mathrm{b}_{\mathcal{Y}}(Y_{s-1}) - \beta \mathrm{b}_{\mathcal{Y}}(Y_s) - \mathrm{r}(X_s)\right),$$

and changing the order of summation gives the result. ■

Theorem 20 suggests the use of Algorithm 1 to compute the gradient of $\sigma^2_{\mathcal{Y}}(\mathrm{b}_{\mathcal{Y}})$ with respect to the parameters of the baseline function $\mathrm{b}_{\mathcal{Y}}$. The theorem implies that, as the number of samples T gets large, the estimate produced by this algorithm approaches the true gradient.

Algorithm 1 Compute estimate of gradient of distance to optimal baseline

given

- A controlled POMDP $(\mathcal{S}, \mathcal{U}, \mathcal{Y}, P, \nu, \mathrm{r}, \mu)$.
- The sequence of states, observations and controls generated by the controlled POMDP, $\{i_0, y_0, u_0, i_1, y_1, \ldots, i_{T-1}, y_{T-1}, u_{T-1}, i_T, y_T\}$.
- A parameterized baseline function $\mathrm{b}_{\mathcal{Y}} : \mathcal{Y} \times \mathbb{R}^L \to \mathbb{R}$.

write $\mathrm{g}^2(y,u)$ to denote $[(\nabla \mu_u(y))/\mu_u(y)]^2$.
set $z_0 = 0$ ($z_0 \in \mathbb{R}^L$), $\Delta_0 = 0$ ($\Delta_0 \in \mathbb{R}^L$)
for all $\{i_t, y_t, u_t, i_{t+1}, y_{t+1}\}$ **do**
  $z_{t+1} = \beta z_t + \nabla \mathrm{b}_{\mathcal{Y}}(y_t,\omega)\mathrm{g}^2(y_t,u_t)$
  $\Delta_{t+1} = \Delta_t + \frac{1}{t+1}\left(\left(\mathrm{b}_{\mathcal{Y}}(y_t,\omega) - \beta \mathrm{b}_{\mathcal{Y}}(y_{t+1},\omega) - \mathrm{r}(x_{t+1})\right) z_{t+1} - \Delta_t\right)$
end for

In Bartlett and Baxter (2002) a variant of the GPOMDP algorithm is shown to give an estimate that, in finite time and with high probability, is close to $\nabla_\beta \eta$. A similar analysis could be performed for the estimate produced by Algorithm 1, in particular, we could replace $\nabla_t = (\nabla \mu_{U_t}(Y_t))/\mu_{U_t}(U_t)$ and $R_t = \mathrm{r}(X_t)$ in Bartlett and Baxter (2002) with $\tilde{\nabla}_t = \mathrm{g}^2(Y_t,U_t)\nabla \mathrm{b}_{\mathcal{Y}}(Y_t)$ and $\tilde{R}_t = \mathrm{b}_{\mathcal{Y}}(Y_{t-1}) - \beta \mathrm{b}_{\mathcal{Y}}(Y_t) - \mathrm{r}(X_t)$. Notice that ∇_t and R_t occur in precisely the same way in GPOMDP to produce an

estimate of

$$\mathbb{E}\left[\nabla_t \sum_{s=t+1}^{\infty} \beta^{s-t-1} R_s\right]$$

as $\tilde{\nabla}_t$ and $\tilde{R}_t$ occur in Algorithm 1 to produce an estimate of

$$\mathbb{E}\left[\tilde{\nabla}_t \sum_{s=t+1}^{\infty} \beta^{s-t-1} \tilde{R}_s\right].$$

Algorithm 2 gives an online version of Algorithm 1. The advantage of such an algorithm is that the baseline may be updated whilst the estimate of the performance gradient is being calculated. Such a strategy for updates would, however, affect the convergence of performance gradient estimates (for constant baselines this may be avoided, see Section 8.2). The question of the convergence of Algorithm 2, and the convergence of performance gradient estimates in the presence of online baseline updates, is not addressed in this paper; though simulations in Sections 8.2 and 8.3 show that performing such online baseline updates can give improvements.

Algorithm 2 Online version of Algorithm 1

given

- A controlled POMDP $(\mathcal{S}, \mathcal{U}, \mathcal{Y}, P, \nu, r, \mu)$.
- The sequence of states, observations and controls generated by the controlled POMDP, $\{i_0, y_0, u_0, i_1, y_1, \ldots, i_{T-1}, y_{T-1}, u_{T-1}, i_T, y_T\}$.
- A parameterized baseline function $b_{\mathcal{Y}} : \mathcal{Y} \times \mathbb{R}^L \to \mathbb{R}$.
- A sequence of step sizes, γ_t

write $g^2(y,u)$ to denote $[(\nabla \mu_u(y))/\mu_u(y)]^2$.
set $z_0 = 0$ $(z_0 \in \mathbb{R}^L)$
for all $\{i_t, y_t, u_t, i_{t+1}, y_{t+1}\}$ **do**
$z_{t+1} = \beta z_t + \nabla b_{\mathcal{Y}}(y_t, \omega_t) g^2(y_t, u_t)$
$\omega_{t+1} = \omega_t - \gamma_t \left(b_{\mathcal{Y}}(y_t, \omega_t) - \beta b_{\mathcal{Y}}(y_{t+1}, \omega_t) - r(x_{t+1})\right) z_{t+1}$
end for

7.2 Minimizing Bound on Squared Error when using a Value Function

Given a controlled MDP and a parameterized class of value functions,

$$\left\{ V(\cdot, \omega) : \mathcal{S} \to \mathbb{R} | \, \omega \in \mathbb{R}^L \right\},$$

we wish to choose a value function to minimize the expected squared error of our gradient estimates. Theorem 18 gives a bound on this error,

$$E_T = \frac{\Omega^*}{T} \mathrm{Var}_\pi\left(\frac{\nabla \mu_u(i)}{\mu_u(i)} V(j, \omega)\right) + \left(\mathbb{E}_\pi\left[\frac{\nabla \mu_u(i)}{\mu_u(i)} A_\beta(j, \omega)\right]\right)^2.$$

We drop ω in the notation, and write $g(i,u)$ to denote $(\nabla \mu_u(i))/\mu_u(i)$, where the gradient is with respect to the policy parameters, θ. We can compute the gradient of this bound:

$$\nabla \frac{1}{2} E_T \quad = \quad \nabla \frac{1}{2}\left(\frac{\Omega^*}{T} \mathrm{Var}_\pi(g(i,u) V(j)) + \left(\mathbb{E}_\pi\left[g(i,u) A_\beta(j)\right]\right)^2\right)$$

$$
\begin{aligned}
&= \frac{1}{2}\left(\frac{\Omega^*}{T}\nabla\mathbb{E}_\pi\left[(\mathrm{g}(i,u)\mathrm{V}(j))^2\right] - \frac{\Omega^*}{T}\nabla\left(\mathbb{E}_\pi\left[\mathrm{g}(i,u)\mathrm{V}(j)\right]\right)^2 + \nabla\left(\mathbb{E}_\pi\left[\mathrm{g}(i,u)\mathrm{A}_\beta(j)\right]\right)^2\right)\\
&= \left(\frac{\Omega^*}{T}\mathbb{E}_\pi\left[(\mathrm{g}(i,u)\mathrm{V}(j))'\left(\mathrm{g}(i,u)\left(\nabla\mathrm{V}(j)\right)'\right)\right]\right.\\
&\quad - \frac{\Omega^*}{T}\left(\mathbb{E}_\pi\left[\mathrm{g}(i,u)\mathrm{V}(j)\right]\right)'\left(\mathbb{E}_\pi\left[\mathrm{g}(i,u)\left(\nabla\mathrm{V}(j)\right)'\right]\right)\\
&\quad \left. - \left(\mathbb{E}_\pi\left[\mathrm{g}(i,u)\mathrm{A}_\beta(j)\right]\right)'\left(\mathbb{E}_\pi\left[\mathrm{g}(i,u)\left(\nabla\mathrm{V}(j)\right)'\right]\right)\right). \qquad (23)
\end{aligned}
$$

This gradient can be estimated from a single sample path of the controlled MDP, $\{X_s, U_s\}$. We need the following assumption on the value function.

Assumption 7. *There are bounds $\mathbf{M}, \mathbf{G} < \infty$ such that for all $i \in \mathcal{S}$, and all $\omega \in \mathbb{R}^L$, the value function is bounded, $|\mathrm{V}(i,\omega)| \leq \mathbf{M}$, and the gradient of the value function is bounded, $\|\nabla\mathrm{V}(i,\omega)\| \leq \mathbf{G}$.*

Algorithm 3 gives an estimate of (23) from a sample path of the controlled MDP; constructing this estimate from the following four estimations:

$$
\begin{aligned}
\Delta A_S &= \frac{1}{S}\sum_{s=0}^{S-1}\left(\mathrm{g}(X_s,U_s)\mathrm{V}(X_{s+1})\right)'\left(\mathrm{g}(X_s,U_s)\left(\nabla\mathrm{V}(X_{s+1})\right)'\right) && \in \mathbb{R}^{1\times L};\\
\Delta B_S &= \frac{1}{S}\sum_{s=0}^{S-1}\mathrm{g}(X_s,U_s)\mathrm{V}(X_{s+1}) && \in \mathbb{R}^{K};\\
\Delta C_S &= \frac{1}{S}\sum_{s=0}^{S-1}\left(\mathrm{r}(X_{s+1}) + \beta\mathrm{V}(X_{s+2}) - \mathrm{V}(X_{s+1})\right)z_{s+1} && \in \mathbb{R}^{K};\\
\Delta D_S &= \frac{1}{S}\sum_{s=0}^{S-1}\mathrm{g}(X_s,U_s)\left(\nabla\mathrm{V}(X_{s+1})\right)' && \in \mathbb{R}^{K\times L},
\end{aligned}
$$

where $z_0 = 0$ and $z_{s+1} = \beta z_s + \mathrm{g}(X_s,U_s)$. The estimate of the gradient then becomes

$$
\Delta_S = \left(\frac{\Omega^*}{T}\Delta A_S - \frac{\Omega^*}{T}\Delta B_S'\Delta D_S - \Delta C_S'\Delta D_S\right)'.
$$

Notice that $\Delta A_S, \Delta B_S$, and ΔD_S are simply sample averages (produced by the Markov chain) estimating the relevant expectations in Equation (23). We see from Theorem 3 and Corollary 5 that the variance of these estimates are $O(\ln(S)/S)$, giving swift convergence. By noting the similarity between the expectation in Equation (20) and the expectation estimated by ΔC_S, we see that the ergodicity and truncation arguments of Theorem 20, and the convergence discussion following, also hold for the ΔC_S estimate.

An online implementation is complicated by the multiplication of expectations. The online algorithm (Algorithm 4) uses a decaying window of time (normalized for the rate of decay) in the calculation of the expectations.

Algorithm 3 Compute estimate of gradient of squared error wrt value function parameter

given

- A controlled POMDP $(\mathcal{S}, \mathcal{U}, \mathcal{Y}, P, \nu, \mathrm{r}, \mu)$.
- The sequence of states, observations and controls generated by the controlled POMDP, $\{i_0, u_0, i_1, \ldots, i_S, u_S, i_{S+1}\}$.
- A parameterized value function $\mathrm{V} : \mathcal{S} \times \mathbb{R}^L \to \mathbb{R}$.

write $\mathrm{g}(i,u)$ to denote $(\nabla \mu_u(i))/\mu_u(i)$.
set $z_0 = 0$ ($z_0 \in \mathbb{R}^K$), $\Delta A_0 = 0$ ($\Delta A_0 \in \mathbb{R}^L$), $\Delta B_0 = 0$ ($\Delta B_0 \in \mathbb{R}^K$), $\Delta C_0 = 0$ ($\Delta C_0 \in \mathbb{R}^K$) and $\Delta D_0 = 0$ ($\Delta D_0 \in \mathbb{R}^{K \times L}$)
for all $\{i_s, u_s, i_{s+1}, i_{s+2}\}$ **do**

$$z_{s+1} = \beta z_s + \mathrm{g}(i_s, u_s)$$
$$\Delta A_{s+1} = \Delta A_s + \tfrac{1}{s+1}\left((\mathrm{g}(i_s,u_s)\mathrm{V}(i_{s+1}))' \left(\mathrm{g}(i_s,u_s)\left(\nabla \mathrm{V}(i_{s+1})\right)'\right) - \Delta A_s\right)$$
$$\Delta B_{s+1} = \Delta B_s + \tfrac{1}{s+1}\left(\mathrm{g}(i_s,u_s)\mathrm{V}(i_{s+1}) - \Delta B_s\right)$$
$$\Delta C_{s+1} = \Delta C_s + \tfrac{1}{s+1}\left(\left(\mathrm{r}(i_{s+1}) + \beta \mathrm{V}(i_{s+2}) - \mathrm{V}(i_{s+1})\right) z_{s+1} - \Delta C_s\right)$$
$$\Delta D_{s+1} = \Delta D_s + \tfrac{1}{s+1}\left(\mathrm{g}(i_s,u_s)\left(\nabla \mathrm{V}(i_{s+1})\right)' - \Delta D_s\right)$$

end for

$$\Delta_S = \left(\tfrac{\Omega^*}{T}\Delta A_S - \tfrac{\Omega^*}{T}\Delta B_S' \Delta D_S - \Delta C_S' \Delta D_S\right)'$$

Algorithm 4 Online version of Algorithm 3

given

- A controlled POMDP $(\mathcal{S}, \mathcal{U}, \mathcal{Y}, P, \nu, \mathrm{r}, \mu)$.
- The sequence of states, observations and controls generated by the controlled POMDP, $\{i_0, u_0, i_1, \ldots, i_S, u_S, i_{S+1}\}$.
- $\alpha \in \mathbb{R}$, $0 < \alpha < 1$
- A sequence of step sizes, γ_s
- A parameterized value function $\mathrm{V} : \mathcal{S} \times \mathbb{R}^L \to \mathbb{R}$.

write $\mathrm{g}(i,u)$ to denote $(\nabla \mu_u(i))/\mu_u(i)$.
set $z_0 = 0$ ($z_0 \in \mathbb{R}^K$), $\Delta A_0 = 0$ ($\Delta A_0 \in \mathbb{R}^L$), $\Delta B_0 = 0$ ($\Delta B_0 \in \mathbb{R}^K$), $\Delta C_0 = 0$ ($\Delta C_0 \in \mathbb{R}^K$) and $\Delta D_0 = 0$ ($\Delta D_0 \in \mathbb{R}^{K \times L}$)
for all $\{i_s, u_s, i_{s+1}, i_{s+2}\}$ **do**

$$z_{s+1} = \beta z_s + \mathrm{g}(i_s, u_s)$$
$$\Delta A_{s+1} = \alpha \Delta A_s + (\mathrm{g}(i_s,u_s)\mathrm{V}(i_{s+1}))' \left(\mathrm{g}(i_s,u_s)\left(\nabla \mathrm{V}(i_{s+1})\right)'\right)$$
$$\Delta B_{s+1} = \alpha \Delta B_s + \mathrm{g}(i_s,u_s)\mathrm{V}(i_{s+1})$$
$$\Delta C_{s+1} = \alpha \Delta C_s + \left(\mathrm{r}(i_{s+1}) + \beta \mathrm{V}(i_{s+2}) - \mathrm{V}(i_{s+1})\right) z_{s+1}$$
$$\Delta D_{s+1} = \alpha \Delta D_s + \mathrm{g}(i_s,u_s)\left(\nabla \mathrm{V}(i_{s+1})\right)'$$
$$\omega_{s+1} = \omega_s - \gamma_s \left(\left(\tfrac{1-\alpha}{1-\alpha^{s+1}}\right)\tfrac{\Omega^*}{T}\Delta A_S - \left(\tfrac{1-\alpha}{1-\alpha^{s+1}}\right)^2 \tfrac{\Omega^*}{T}\Delta B_S' \Delta D_S - \left(\tfrac{1-\alpha}{1-\alpha^{s+1}}\right)^2 \Delta C_S' \Delta D_S\right)'$$

end for

7.3 Minimizing the Bias Error when using a Value Function

As the number of steps T gets large, the error E_T of the gradient estimate becomes proportional to the square of the bias error,

$$E_\infty = \left(\mathbb{E}_\pi\left[\mathrm{g}(i,u)\mathrm{A}_\beta(j)\right]\right)^2.$$

The gradient of this quantity, with respect to the parameters of the value function, can be computed using Algorithm 3, with $\Omega^*/T = 0$. In this case, only ΔC_s and ΔD_s need to be computed.

7.4 Minimizing Bound on Sample Error when using a Value Function

A more restrictive approach is to minimize the error seen at each sample,

$$R = \mathbb{E}_\pi\left(\mathrm{g}(i,u)\mathrm{A}_\beta(j)\right)^2.$$

This approach directly drives V towards J_β and as such does not aim for additional beneficial correlation. It produces an algorithm that is very similar to TD Sutton (1988), but has the benefit that the relative magnitude of the gradient with respect to the policy parameters is taken into account. In this way, more attention is devoted to accuracy in regions of the state space with large gradients.

For a parameterized class of value functions, $\left\{\mathrm{V}(\cdot,\omega) : \mathcal{S} \to \mathbb{R} | \, \omega \in \mathbb{R}^L\right\}$, we can determine the gradient of this quantity.

$$\begin{aligned}
\nabla\frac{1}{2}R &= \nabla\frac{1}{2}\mathbb{E}_\pi\left(\mathrm{g}(i,u)\mathrm{A}_\beta(j)\right)^2 \\
&= -\mathbb{E}_\pi\left[\left(\mathrm{g}(i,u)\left(\nabla\mathrm{V}(j)\right)'\right)'\left(\mathrm{g}(i,u)\mathrm{A}_\beta(j)\right)\right] \\
&= -\mathbb{E}_\pi\left[\left(\mathrm{g}(i,u)\right)^2\nabla\mathrm{V}(j)\mathrm{A}_\beta(j)\right].
\end{aligned}$$

If the value function satisfies Assumption 7, the gradient may be estimated by a single sample path from a controlled MDP. The ergodicity and truncation argument is the same as that in the proof of Theorem 20.

$$\Delta R_T = \frac{1}{T}\sum_{t=1}^{T}\left(\mathrm{r}(X_t) + \beta\mathrm{V}(X_{t+1}) - \mathrm{V}(X_t)\right)z_t,$$

where $z_0 = 0$, and $z_{t+1} = \beta z_t + \left(\mathrm{g}(X_t,U_t)\right)^2\nabla\mathrm{V}(X_{t+1})$.

8. Simulation Examples

This section describes some experiments performed in simulated environments. First, the estimates suggested by Sections 5 and 6 are tested in a simple, simulated setting. This simple setting is then used to test the algorithms of Section 7. Finally, a larger, target-tracking setting is used to test a number of gradient estimates at various stages of the learning process.

8.1 Three State MDP, using Discounted Value Function

This section describes experiments comparing choices of control variate for a simple three state MDP. The system is the described in detail in Baxter et al. (2001). The gradient $\nabla\eta$ was compared to the gradient estimates produced with a variety of schemes: GPOMDP without any control variate; a constant baseline set to $\mathbb{E}_\pi\mathrm{J}_\beta(i)$; the optimum constant baseline, described in Theorem 11; the

optimum baseline function, described in Corollary 15; and a value function that was trained using Algorithm 3 with Ω^*/T set to 0.001. This value function had a distinct parameter for each state, all initially set to zero.

Because of its simplicity, a number of quantities can be computed explicitly, including the true gradient $\nabla\eta$, the discounted value function J_β, the expectation of the discounted value function, the optimal baseline, and the optimal constant baseline. All experiments used the precomputed discounted value function in their $\nabla_\beta\eta$ estimations rather than the discounted sum of future rewards, an estimate of the discounted value function. For each experiment, the data was collected over 500 independent runs, with $\beta = 0.95$.

Figures 2 and 3 plot the mean and standard deviation (respectively) of the relative norm difference of the gradient estimate from $\nabla\eta$, as a function of the number of time steps. The relative norm difference of a gradient estimate Δ from the true gradient $\nabla\eta$ is given by

$$\frac{\|\Delta - \nabla\eta\|}{\|\nabla\eta\|}.$$

It is clear from the figures that the use of these control variates gives significant variance reductions over GPOMDP. It is also clear that the optimum baseline gives better performance than the use of the expectation of the discounted value function as a baseline. For this MDP, the performance difference between the optimum baseline and the optimum constant baseline is small; the optimum baseline of this system, $b^*_{\mathcal{Y}} = (6.35254, 6.35254, 6.26938)'$, is close to a constant function. The optimum constant baseline is $b^* = 6.33837$.

Since the value of Ω^*/T was fixed when optimizing the value function, the asymptotic error of its associated gradient estimate is non-zero, as Figure 2 shows. However, the expected error remains smaller than that of GPOMDP for all but very large values of T, and the standard deviation is always smaller.

8.2 Online Training

Instead of precomputing the optimum baseline, and pretraining the value function, they could be learned online, whilst estimating $\nabla_\beta\eta$. Figures 4 and 5 show experiments on the same three state MDP as in Section 8.1, but here the baseline and value function were learned online, using Algorithm 2 and Algorithm 4 respectively. GPOMDP and baseline plots were over 500 independent runs, the value function plots were over 1000 independent runs. A β value of 0.95 was used, and the online training step size γ_t was set to $1/\ln(1+t)$. For the value function, Ω^*/T was set to 0.01 and α was set to 0.99. The baseline and the value function had a parameter for each state and were initially set to zero.

It is clear from the figures that the online baseline algorithm gives a significant improvement from the GPOMDP algorithm. Looking at the error using the online value function algorithm we see a performance increase over GPOMDP until T becomes large.

Note that the baseline, when trained online, is non-stationary, and the gradient estimate becomes

$$\Delta = \frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)} \left(\sum_{s=t+1}^{T} \beta^{s-t-1} r(X_s) - b_t(Y_t) \right).$$

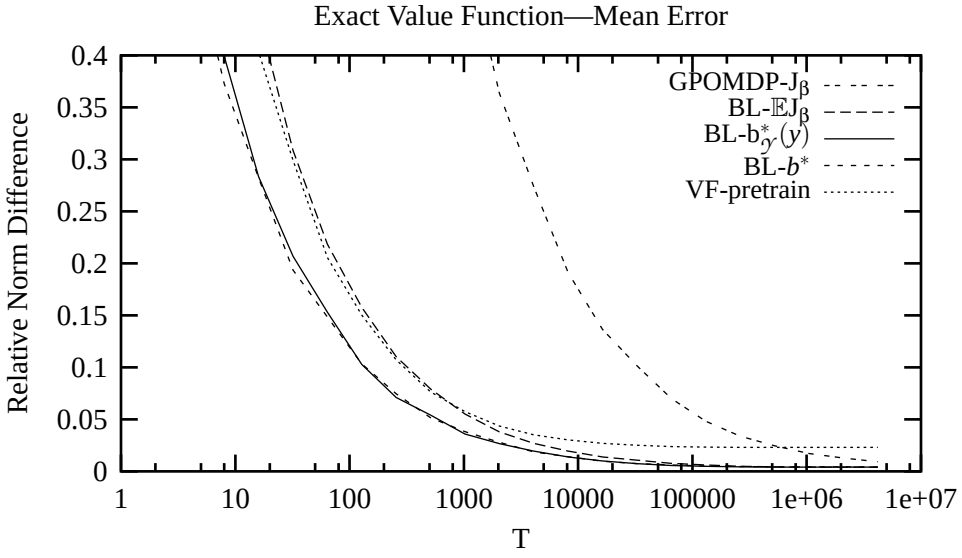


Figure 2: The mean of the relative norm difference from $\nabla\eta$: using no control variate (GPOMDP-J_β); using the expected discounted value function as a baseline (BL-$\mathbb{E}J_\beta$); using the optimum baseline (BL-$b^*_\gamma(y)$); using the optimal constant baseline (BL-b^*); and using a pretrained value function (VF-pretrained). In all cases the explicitly calculated discounted value function was used in place of the estimates J_t (except, of course, for the pretrained value function case, where the value function is used in place of the estimates J_t.)

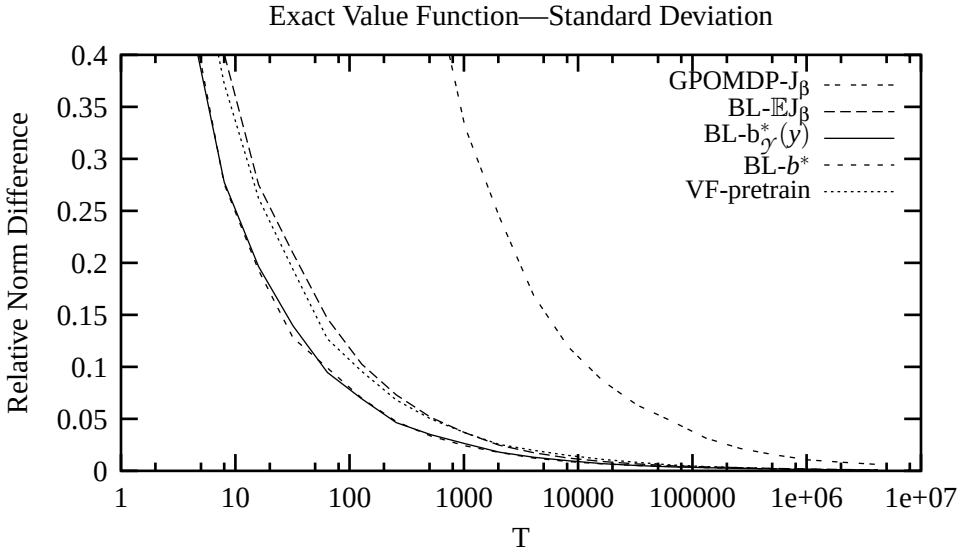


Figure 3: The standard deviation of the relative norm difference from $\nabla\eta$ (see Figure 2 for an explanation of the key).

This non-stationarity could mean an additional bias in the estimate, though we can see by the graphs that, at least in this case, this additional bias is small. The estimate that we actually use is

$$\Delta = \frac{1}{T}\sum_{t=1}^{T} z_t \left(\mathrm{r}(X_t) - (\mathrm{b}_{t-1}(Y_{t-1}) - \beta \mathrm{b}_{t-1}(Y_t))\right),$$

where z_t, the eligibility trace, is given by $z_0 = 0$ and $z_{t+1} = \beta z_t + (\nabla \mu_{U_t}(Y_t))/\mu_{U_t}(Y_t)$. One might argue that this additionally correlates our baseline with any errors due to the truncation of the sum of discounted future rewards. This should make little difference, except for small T; we have seen that, for the modified estimate $\Delta_T^{(+S)}(\mathrm{b}_\mathcal{Y})$, any influence this error has is exponentially decreasing.

Note that for any constant baseline we need not worry about non-stationarity, as we have

$$\sum_{t=1}^{T} z_t (b_T - \beta b_T) = \left(\sum_{t=1}^{T} z_t\right)(1-\beta) b_T,$$

so by additionally keeping track of $\Sigma_T = \sum_{t=1}^{T} z_t$ we have the estimate, at time T,

$$\frac{1}{T}\sum_{t=1}^{T} z_t \left(\mathrm{r}(X_t) - (b_T - \beta b_T)\right) = \frac{1}{T}\sum_{t=1}^{T} z_t \mathrm{r}(X_t) - \frac{1}{T}\Sigma_T (1-\beta) b_T,$$

an unbiased estimate of $\nabla_\beta \eta$; again, treating the error due to the truncation of the discounted sum of future rewards as negligible.

8.3 Locating a Target

These experiments deal with the task of a puck, moving in a plane, learning to locate a target. The puck had unit mass, 0.05 unit radius, and was controlled by applying a 5 unit force in either the positive or negative x direction and either the positive or negative y direction. The puck moved within a 5×5 unit area with elastic walls and a coefficient of friction of 0.0005; gravity being set to 9.8. The simulator worked at a granularity of $1/100$ of a second with controller updates at every $1/20$ of a second. The distance between the puck and the target location was given as a reward at each update time. Every 30 seconds this target and the puck was set to a random location, and the puck's x and y velocities set randomly in the range $[-10, 10]$.

The puck policy was determined by a neural network with seven inputs, no hidden layer, and four outputs; the outputs computing a tanh squashing function. The inputs to the controller were: the x and y location of the puck, scaled to be in $[-1, 1]$; the x and y location of the puck relative to the target, scaled by the dimension sizes; the velocity of the puck, scaled such that a speed of 10 units per second was mapped to a value of 1; and a constant input of 1 to supply an offset. The outputs of the neural network gave a weighting, $\xi_i \in (0, 1)$, to each of the (x, y) thrust combinations: $(-5, -5)$; $(-5, 5)$; $(5, -5)$; and $(5, 5)$. So, collating the seven inputs in the vector v, we have

$$\xi_i = \mathrm{sqsh}\left(\sum_{k=1}^{7} \theta_{i,k} v_k\right), \qquad i \in \{1, 2, 3, 4\},$$

where θ is a vector of 28 elements, one element, $\theta_{i,k}$, for each i, k pair, and the squashing function is $\mathrm{sqsh}(x) = (1 + \tanh(x))/2$. The probability of the i^{th} thrust combination is then given by

$$\mu_i(v, \theta) = \frac{\xi_i}{\sum_j \xi_j},$$

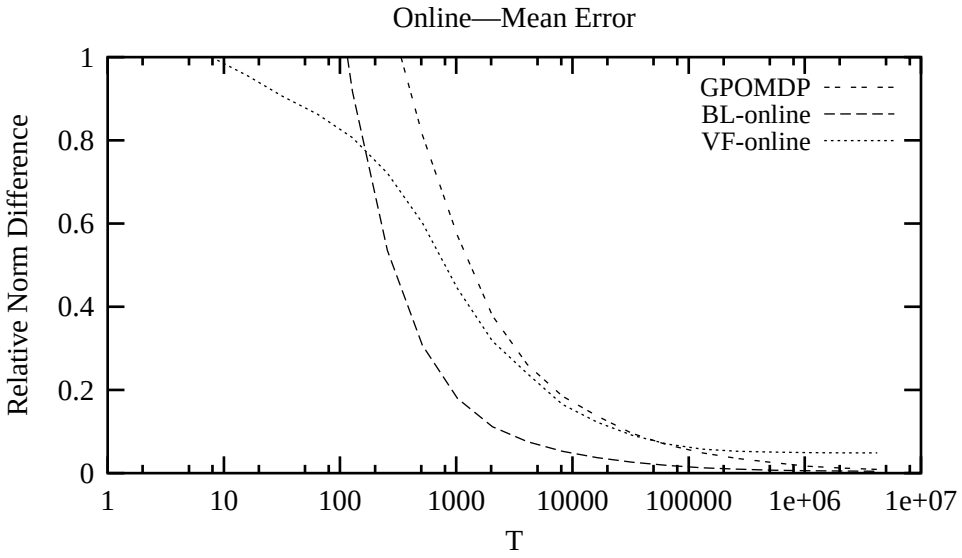


Figure 4: The mean of the relative norm difference from $\nabla\eta$: using no control variate (GPOMDP); using a baseline trained online (BL-online); and using a value function trained online (VF-online).

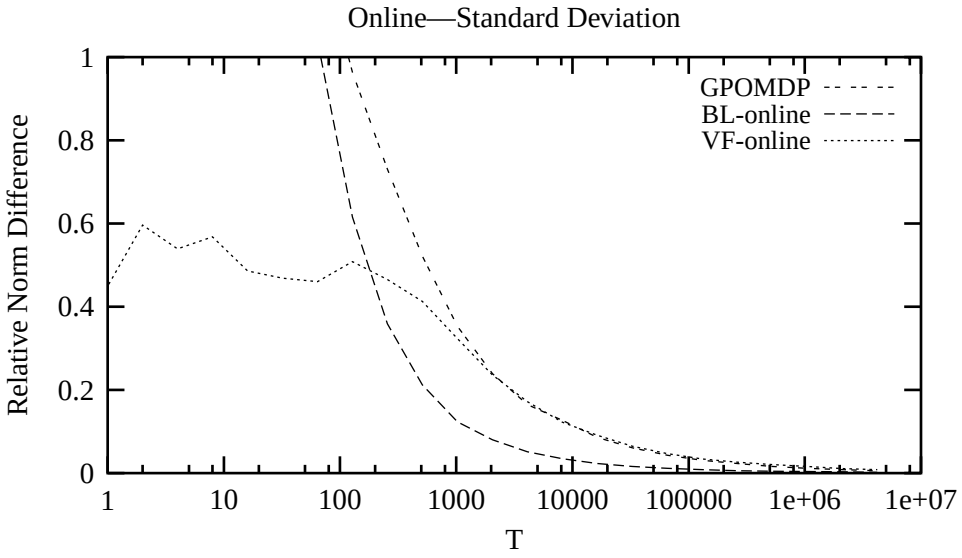


Figure 5: The standard deviation of the relative norm difference from $\nabla\eta$ (see Figure 4 for an explanation of the key).

where actions have been labelled with the associated thrust combination.

The puck was first trained using conjugate gradient ascent, with GPOMDP (with $T = 10^8$) used to estimate the gradient. The parameters of the policy were recorded at 28 points along the training curve: at the initial parameter setting ($\theta = 0$); and at each change in line search direction. Results for 4 of the 28 points are shown in Figure 6. The results show the mean, over 100 independent trials, of the relative norm difference between gradient estimates when using a range of different baselines, all learned online ($\gamma_t = 1/\ln(1+t)$) and initially set to zero, and an estimate of $\nabla_\beta \eta$. The second order baseline was a second order polynomial of the inputs, that is, again collating the inputs in the vector v,

$$b(v,\omega) = \omega_{0,0} + \sum_{k=1}^{7} \omega_{k,0} v_k + \sum_{k=1}^{7} \sum_{l=k}^{7} \omega_{k,l} v_k v_l,$$

where ω is a vector of 32 elements, with one element, $\omega_{k,l}$, for each second order term $v_k v_l$, one additional element, $\omega_{k,0}$, for each first order term v_k, and one additional element, $\omega_{0,0}$, for the constant term. The estimate of $\nabla_\beta \eta$ was produced by averaging the unbiased $\nabla_\beta \eta$ estimates at $T = 2^{23}$; an average over 400 samples.

Figure 6 shows that each baseline method performed better than GPOMDP, with the second order baseline performing the best of these. The estimated average reward and the estimated optimal constant baseline performed almost equally, and both performed better than the online constant baseline in this case. That the two estimation methods performed almost equally would suggest that, in this case, the random variables $(\nabla \mu_u(y)/\mu_u(y))^2$ and $J_\beta(j)$ are close to independent. It might be that for most policies, or at least policies at the θ values we tested, $\|\mathbb{E}_\pi J_\beta(i)\| \gg \|J_\beta(i) - \mathbb{E}_\pi J_\beta(i)\|$, since this implies

$$\begin{aligned}
\mathbb{E}_\pi \left[\left(\frac{\nabla \mu_u(y)}{\mu_u(y)} \right)^2 J_\beta(j) \right] &= \mathbb{E}_\pi \left(\frac{\nabla \mu_u(y)}{\mu_u(y)} \right)^2 \mathbb{E}_\pi J_\beta(i) + \mathbb{E}_\pi \left[\left(\frac{\nabla \mu_u(y)}{\mu_u(y)} \right)^2 \left(J_\beta(j) - \mathbb{E}_\pi J_\beta(i) \right) \right] \\
&\approx \mathbb{E}_\pi \left(\frac{\nabla \mu_u(y)}{\mu_u(y)} \right)^2 \mathbb{E}_\pi J_\beta(i).
\end{aligned}$$

9. Conclusions

We have shown that the use of control variate techniques can reduce estimation variance when estimating performance gradients. The first technique was to add a baseline. Here we analyzed the variance quantities of (8) and (9), the estimation variance when using a baseline under the assumption that the discounted value function is known and samples may be drawn independently. We have given the optimal baseline, the baseline that minimizes this variance, and have expressed the additional variance resulting from using an arbitrary baseline as a weighted squared distance from this optimum. Similar results have also been shown for a constant baseline. Here it was also shown how much additional variance results from using the expected discounted value function, a popular choice of baseline, in place of the optimal constant baseline. We have also shown that the estimation variance from $\Delta_T^{(+S)}(b_\mathcal{Y})$, a realizable estimate of $\nabla_\beta \eta$ formed from a single sample path of the associated POMDP, is bounded by the stationary variance plus a term independent of the choice of baseline, and another term of negligible magnitude.

A second control variate technique used to reduce estimation variance was to replace estimates of the discounted value function with some appropriate value function V. We have shown that, even

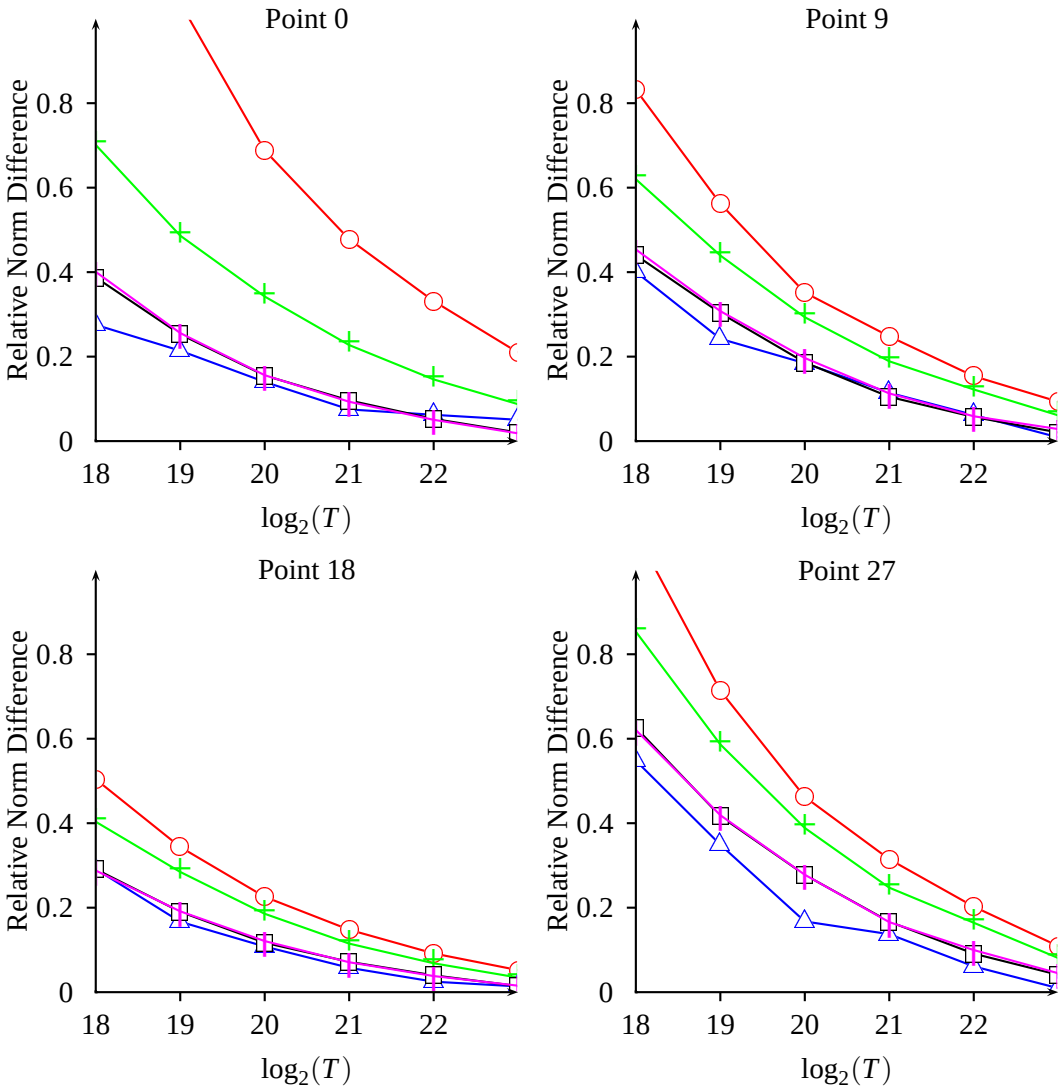


Figure 6: Each plot shows the mean, over 100 independent runs, of the relative norm difference from $\nabla_\beta \eta$: using no baseline ( ); using a constant baseline, trained online ( ); using a second order polynomial of the inputs as a baseline, trained online ( ); using $\mathbb{E}_\pi J_\beta(i)$ as a baseline, estimated online ( ); and using the optimal constant baseline, estimated online ( ). The reference $\nabla_\beta \eta$ is estimated by averaging the unbiased estimates at $T = 2^{23}$. The four plots show four of the 28 parameter values at the end points of each line search in the conjugate gradient ascent algorithm, when training on the target location example using GPOMDP (with $T = 10^8$) to produce gradient estimates. The remaining 24 parameter values give similar plots.

if the discounted value function is known, selecting V to be equal to the discounted value function is not necessarily the best choice. We have shown examples where this is the case; where additional reduction in estimation variance can be achieved by selecting V to be a function other than the discounted value function, with no addition of estimation bias. We have also given a bound on the expected squared error of the estimate Δ_T^{V}, an estimate of $\nabla_\beta \eta$ that uses V in place of discounted value function estimates and is formed from a single sample path of the associated MDP.

The gradient estimates $\Delta_T^{(+S)}(\mathrm{b}_\mathcal{Y})$ and Δ_T^{V} use a baseline $\mathrm{b}_\mathcal{Y}$ and a value function V, respectively, in their calculations. In experiments on a toy problem we investigated the improvements obtained when using the optimal choice of baseline in $\Delta_T^{(+S)}(\mathrm{b}_\mathcal{Y})$, and also when using the value function minimizing the bound on expected squared error of estimates in Δ_T^{V}. Significant improvement was shown.

In general the optimal choices for the baseline and the value function may not be known. We have explored the idea of using gradient descent on the error bounds derived in this paper to learn a good choice for a baseline, or for a value function. We have given realizable algorithms to obtain the appropriate gradient estimates, along with their online versions. In experiments on the toy problem, and in a target location problem, we have seen some improvements given by these algorithms.

In experiments we have looked at using the online versions of the algorithms in Section 7; updating the baseline (or value function) whilst estimating the gradient of the performance. Consequently some additional bias in the performance gradient estimate is likely to have occurred. The results of the experiments, however, would suggest that this bias is small. Further work of interest is the study of the convergence of these online algorithms, and also the convergence of the performance whilst using these online algorithms.

Acknowledgments

Most of this work was performed while the authors were with the Research School of Information Sciences and Engineering at the Australian National University, supported in part by the Australian Research Council.

Appendix A. Discussion of Assumption 1

The details in this section can be found in texts covering Markov chains; see, for example, Puterman (1994); Grimmett and Stirzaker (1992); Seneta (1981). We include them to: define the terms used in Assumption 1; show how Assumption 1 may be relaxed; and give an intuition of our use of Assumption 1.

The states of a Markov chain $M = (\mathcal{S}, P)$ can be divided into equivalence classes under the communicating relation $\leftrightarrow$. We define $i \leftrightarrow i$, and write $i \leftrightarrow j$ if there are integers $m, n > 0$ such that $p_{ij}^{(m)} > 0$ and $p_{ji}^{(n)} > 0$, where $p_{ij}^{(t)}$ is the ij^{th} entry of the t-step transition matrix P^t. We call a class $\tilde{\mathcal{S}} \subset \mathcal{S}$ recurrent if its states are recurrent, otherwise we call it transient. A state is recurrent if $\Pr\{X_t = i \text{ for some } t > 0 | X_0 = i\} = 1$, otherwise it is transient. Notice that this means that once the chain enters a recurrent class it never leaves, but rather visits all states of that class infinitely often. If the chain is finite then it will eventually leave every transient class and settle in some recurrent class.

We say a Markov chain $M = (\mathcal{S}, P)$ is irreducible if the space $\mathcal{S}$ forms a single class under $\leftrightarrow$; necessarily a recurrent class for finite Markov chains. We can relax the irreducibility condition, and instead allow any $\mathcal{S}$ that contains a single recurrent class $\mathcal{S}_R$ plus a set (possibly containing more than one class) of transient states $\mathcal{S}_T$ such that $\Pr\{X_t \in \mathcal{S}_R \text{ for some } t > 0 | X_0 = j\} = 1$ for all $j \in \mathcal{S}_T$ (guaranteed for finite chains).

The period, d, of a state $i \in \mathcal{S}$ of a Markov chain $M = (\mathcal{S}, P)$ is the greatest common divisor of the set of times $\{t > 0 : p_{ii}^{(t)} > 0\}$. It is uniform across the states of a class. A state, and consequently a class, is aperiodic if $d = 1$. We can relax the aperiodicity condition and allow arbitrary periods. Consider $\mathcal{S}_R$ to be constructed $\mathcal{S}_R = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{d-1}$, where d is the period of $\mathcal{S}_R$ and the sets $\mathcal{S}_k$ are chosen such that $\Pr\{X_{t+1} \in \mathcal{S}_{k+1 (\text{mod } d)} | X_t \in \mathcal{S}_k\} = 1$.

Our interest is in the existence and uniqueness of the stationary distribution π. The existence of π stems from the Markov chain reaching, and never leaving, a recurrent class, combined with the forgetfulness of the Markov property. The uniqueness of π stems from us allowing only a single recurrent class. So given a finite Markov chain $M = (\mathcal{S}, P)$ with the construction $\mathcal{S} = \mathcal{S}_T \cup \mathcal{S}_R$, and $\mathcal{S}_R = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{d-1}$, as above, we have, writing $N_{[0,T)}(i)$ to denote the number of times state i is visited before time T,

$$\lim_{T \to \infty} T^{-1} N_{[0,T)}(i) = \pi_i, \qquad \text{a.s.} \tag{24}$$

Equation (24) is helpful in two ways. Firstly, our choice of performance measure, (2), is the expected average of $\mathrm{r}(X_t)$, where $\{X_t\}$ is produced by the chain. We see from (24) that this value is independent of the initial state, and we could equivalently use the expected value of $\mathrm{r}(X)$, with $X \sim \pi$. Secondly, we are interested in calculating expectations over the stationary distribution (such as $\nabla_\beta \eta$), and we see from (24) that this expectation can be calculated by observing a single sample path generated by the Markov chain, almost surely. In Section 4 it is seen that we can even do well with a finite length sample path; it is here we use the assumption of irreducibility and aperiodicity.

The analytical results of Section 5 and Section 6 use Theorem 3, Theorem 4 and Corollary 5 of Section 4 to bound the variance terms of the form

$$\mathrm{Var}\left(\frac{1}{T} \sum_{t=0}^{T-1} \mathrm{f}(X_t) \right), \tag{25}$$

where X_t is generated by a Markov chain $M = (\mathcal{S}, P)$ starting in the stationary distribution $X_0 \sim \pi$, with variance terms of the form

$$\mathrm{Var}(\mathrm{f}(X)), \tag{26}$$

where $X \sim \pi$. The proofs of these results use the property

$$\lim_{T \to \infty} \Pr\{X_T = i\} = \pi_i, \tag{27}$$

which holds when $\mathcal{S}_R$ is aperiodic, and is stronger than Equation (24). In particular, Equation (27) holds with the addition of the set of transient states $\mathcal{S}_T$; indeed the variance quantities of Equation (25) and (26) are not affected by such an addition, as the set $\mathcal{S}_T$ has π-measure zero. Also, when $\mathcal{S}_R$ is periodic, writing $X_t^{(k)}$ for the d-step subprocess with elements in $\mathcal{S}_k$ and $\pi^{(k)}$ for the

stationary distribution corresponding to this irreducible aperiodic chain, we have

$$\begin{aligned}
&\mathrm{Var}\left(\frac{1}{dT}\sum_{t=0}^{dT-1}\mathrm{f}(X_t)\right)\\
&= \mathbb{E}\left[\left(\frac{1}{dT}\sum_{t=0}^{dT-1}\mathrm{f}(X_t)-\mathbb{E}\left[\frac{1}{dT}\sum_{t=0}^{dT-1}\mathrm{f}(X_t)\middle| X_0\sim\pi\right]\right)^2\middle| X_0\sim\pi\right]\\
&= \frac{1}{d^2}\sum_{k_1=0}^{d-1}\sum_{k_2=0}^{d-1}\mathbb{E}\left[\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k_1)}\right)-\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k_1)}\right)\middle| X_0\sim\pi\right]\right)\right.\\
&\qquad\left.\times\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k_2)}\right)-\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k_2)}\right)\middle| X_0\sim\pi\right]\right)\middle| X_0\sim\pi\right]\\
&\leq \frac{2}{d}\sum_{k=0}^{d-1}\mathbb{E}\left[\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k)}\right)-\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k)}\right)\middle| X_0\sim\pi\right]\right)^2\middle| X_0\sim\pi\right]\\
&= \frac{2}{d}\sum_{k=0}^{d-1}\mathbb{E}\left[\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k)}\right)-\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k)}\right)\middle| X_0^{(k)}\sim\pi^{(k)}\right]\right)^2\middle| X_0^{(k)}\sim\pi^{(k)}\right]\\
&= \frac{2}{d}\sum_{k=0}^{d-1}\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}\left(X_t^{(k)}\right)\right)
\end{aligned}$$

(that the distribution of $X_0^{(k)}$ is $\pi^{(k)}$ when $X_0\sim\pi$ is due to the sets $\mathcal{S}_k$ having equal π-measure.) It is now straightforward to give analogous results to those of Section 5 and 6 when the Markov chain consists of a single, possibly periodic, recurrent class plus a set of transient states.

The justification in studying the variance quantity (25) is that, after leaving the set of states $\mathcal{S}_T$, the distribution over states will approach π exponentially quickly. Whilst this does not hold for periodic chains, it does hold that the distribution over states restricted to the set of times $\{T_k+dt : t\in\{0,1,2\ldots\}\}$, where T_k is the first time X_t hits the set S_k, will approach $\pi^{(k)}$ exponentially quickly.

Appendix B. Proofs for Section 4

In this section we give the proofs for Theorem 2, Theorem 3, Theorem 4, and Corollary 5 of Section 4. Before giving the proof of Theorem 2 we first look at some properties of Markov chains. In particular, we look at the covariance decay matrix of a finite ergodic Markov chain.

Definition 4. *Let $M=(\mathcal{S},P)$ be a finite ergodic Markov chain, and let π be its stationary distribution. We denote the* covariance decay matrix *of this chain by $D(t)$, and define it by*

$$D(t)\stackrel{\text{def}}{=}\Pi^{\frac{1}{2}}\left(P^t-e\pi'\right)\Pi^{-\frac{1}{2}}$$

where, given $\mathcal{S}=\{1,2,\ldots,n\}$, $\Pi^{\frac{1}{2}}=\mathrm{diag}(\sqrt{\pi_1},\sqrt{\pi_2},\ldots,\sqrt{\pi_n})$, and $\Pi^{-\frac{1}{2}}=\left[\Pi^{1/2}\right]^{-1}$.

We will see that the gain of the auto-covariance over the variance can be bound by the spectral norm of the covariance decay matrix. First we will give a bound on the spectral norm of the covariance decay matrix for general finite ergodic Markov chains. Then we will give a much tighter

bound for reversible finite ergodic Markov chains. The spectral norm of a matrix is given by the following definition.

Definition 5. *The* spectral norm *of a matrix A is denoted* $\|A\|_\lambda$. *It is the matrix norm induced by the Euclidean norm,*

$$\|A\|_\lambda \stackrel{\text{def}}{=} \max_{\|x\|=1} \|Ax\|.$$

An equivalent definition is

$$\|A\|_\lambda = \max_{\|x\|=1} \|Ax\| = \max_{\|x\|=1} \sqrt{x'A'Ax} = \sqrt{\lambda_{\max}(A'A)},$$

where $\lambda_{\max}(A)$ *denotes the largest eigenvalue of the matrix A. As* $A'A$ *is symmetric and positive semi-definite, all of its eigenvalues are real and positive.*

Note 2. *We have that, for any matrix A*

$$\|Ax\| \leq \|A\|_\lambda \|x\|.$$

This can be seen from: for $\|x\| \neq 0$

$$\|Ax\| = \left\| A\left(\frac{x}{\|x\|}\right) \right\| \|x\|.$$

Recall the following two definitions.

Definition 6. *The* total variation distance *between two distributions* p, q *on the finite set* $\mathcal{S}$ *is given by*

$$\mathrm{d}_{TV}(p,q) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{i\in\mathcal{S}} |p_i - q_i| = \sum_{i\in\mathcal{S}: p_i > q_i} (p_i - q_i).$$

Definition 7. *The* mixing time *of a finite ergodic Markov chain* $M = (\mathcal{S}, P)$ *is defined as*

$$\tau \stackrel{\text{def}}{=} \min\left\{ t > 0 : \max_{i,j} \mathrm{d}_{TV}\left(P_i^t, P_j^t\right) \leq e^{-1} \right\},$$

where P_i^t *denotes the* i^{th} *row of the t-step transition matrix* P^t.

Note 3. *Denoting* $d_t \stackrel{\text{def}}{=} \max_{i,j} \mathrm{d}_{TV}\left(P_i^t, P_j^t\right)$, *for* $s,t \geq 1$ *we have that* $d_{t+s} \leq d_t d_s$, *and hence*

$$d_t \leq \exp\left(-\lfloor t/\tau \rfloor\right).$$

Note 4. *We have that*

$$\max_{i\in\mathcal{S}} \mathrm{d}_{TV}\left(P_i^t, \pi\right) \leq d_t.$$

The sub-multiplicative property in Note 3 can be seen from

$$\begin{aligned}
\mathrm{d}_{TV}\left(P_i^{t+s},P_j^{t+s}\right) &= \sum_{l\in\mathcal{S}:p_{il}^{(t+s)}>p_{jl}^{(t+s)}}\left(p_{il}^{(t+s)}-p_{jl}^{(t+s)}\right)\\
&= \sum_{l\in\mathcal{S}:p_{il}^{(t+s)}>p_{jl}^{(t+s)}}\sum_{k\in\mathcal{S}}\left(p_{ik}^{(t)}-p_{jk}^{(t)}\right)p_{kl}^{(s)}\\
&= \sum_{k\in\mathcal{S}:p_{ik}^{(t)}>p_{jk}^{(t)}}\left(p_{ik}^{(t)}-p_{jk}^{(t)}\right)\sum_{l\in\mathcal{S}:p_{il}^{(t+s)}>p_{jl}^{(t+s)}}p_{kl}^{(s)}\\
&\quad -\sum_{k\in\mathcal{S}:p_{jk}^{(t)}>p_{ik}^{(t)}}\left(p_{jk}^{(t)}-p_{ik}^{(t)}\right)\sum_{l\in\mathcal{S}:p_{il}^{(t+s)}>p_{jl}^{(t+s)}}p_{kl}^{(s)}\\
&\leq d_{TV}\left(P_i^t,P_j^t\right)\max_{k_1,k_2\in\mathcal{S}}\sum_{l\in\mathcal{S}:p_{il}^{(t+s)}>p_{jl}^{(t+s)}}\left(p_{k_1l}^{(s)}-p_{k_2l}^{(s)}\right)\\
&\leq d_{TV}\left(P_i^t,P_j^t\right)d_s, \qquad (28)
\end{aligned}$$

where $p_{ij}^{(t)}$ denotes the ij^{th} component of P^t, and we have used that $\sum_l p_{il}^{(t)}p_{lk}^{(s)}=p_{ik}^{(t+s)}$. As $d_t\leq 1$, this also implies d_t is non-increasing (for $t\geq 1$). The inequality in Note 3 then follows from applying the sub-multiplicative property to $\tau\lfloor t/\tau\rfloor\leq t$, giving

$$d_t\leq\begin{cases} d_\tau^{\lfloor t/\tau\rfloor} & t\geq\tau,\\ 1 & t<\tau.\end{cases}$$

Note 4 can be seen from

$$\sum_{k\in\mathcal{S}}\left|p_{ik}^{(t)}-\pi_k\right|=\sum_{k\in\mathcal{S}}\left|\sum_{j\in\mathcal{S}}\pi_j\left(p_{ik}^{(t)}-p_{jk}^{(t)}\right)\right|\leq\sum_{j\in\mathcal{S}}\pi_j\sum_{k\in\mathcal{S}}\left|p_{ik}^{(t)}-p_{jk}^{(t)}\right|.$$

We will also consider the following, asymmetric, notion of distance.

Definition 8. *The* χ^2 distance *between the distribution* p *and the distribution* q *on the finite set* $\mathcal{S}$, *with* $q_i>0$ *for all* $i\in\mathcal{S}$, *is given by*

$$\mathrm{d}_{\chi^2}(p,q)\stackrel{\text{def}}{=}\left(\sum_{i\in\mathcal{S}}\frac{(p_i-q_i)^2}{q_i}\right)^{1/2}.$$

Lemma 21. *Let* $M=(\mathcal{S},P)$ *be a finite ergodic Markov chain, and let* π *be its stationary distribution. There exists a mixing time* τ, *which is a property of* M, *such that*

$$\|D(t)\|_\lambda\leq\sqrt{\mathbb{E}_{i\sim\pi}\left(\mathrm{d}_{\chi^2}(P_i^t,\pi)\right)^2}\leq\sqrt{2\left|\mathcal{S}\right|\max_{i\in\mathcal{S}}\mathrm{d}_{TV}(P_i^t,\pi)}\leq\sqrt{2\left|\mathcal{S}\right|\exp\left(-\lfloor t/\tau\rfloor\right)}.$$

Thus we have $\|D(t)\|_\lambda\leq\sqrt{2\left|\mathcal{S}\right|\exp\left(-\lfloor t/\tau\rfloor\right)}$.

Proof. Note that $D(t)'D(t)$ is symmetric and positive semi-definite and hence its eigenvalues are real and positive. Label them, in non-increasing order, $\lambda_1, \lambda_2, \ldots$. This combined with the relationship $\sum_i \lambda_i = \mathrm{tr}\left(D(t)'D(t)\right)$, where $\mathrm{tr}(A)$ denotes the trace of the matrix A, gives

$$0 \leq \lambda_1 \leq \mathrm{tr}\left(D(t)'D(t)\right).$$

Furthermore,

$$\begin{aligned}
\mathrm{tr}\left(D(t)'D(t)\right) &= \sum_{i\in\mathcal{S}}\sum_{k\in\mathcal{S}} \frac{\pi_i}{\pi_k}\left(p_{ik}^{(t)} - \pi_k\right)^2 = \mathbb{E}_{i\sim\pi}\left(\mathrm{d}_{\chi^2}\left(P_i^t, \pi\right)\right)^2 \\
&= \sum_{i\in\mathcal{S}}\sum_{k\in\mathcal{S}} \frac{\pi_i p_{ik}^{(t)}}{\pi_k}\left(p_{ik}^{(t)} - \pi_k\right) - \sum_{i\in\mathcal{S}}\sum_{k\in\mathcal{S}} \pi_i\left(p_{ik}^{(t)} - \pi_k\right) \\
&\leq \sum_{k\in\mathcal{S}} \max_{\tilde{i}\in\mathcal{S}} \left|p_{\tilde{i}k}^{(t)} - \pi_k\right| \left(\frac{1}{\pi_k}\sum_{i\in\mathcal{S}} \pi_i p_{ik}^{(t)}\right) \\
&\leq |\mathcal{S}| \max_{i\in\mathcal{S}} \sum_{k\in\mathcal{S}} \left|p_{ik}^{(t)} - \pi_k\right| \\
&= 2\,|\mathcal{S}| \max_{i\in\mathcal{S}} \mathrm{d}_{TV}\left(P_i^t, \pi\right) \\
&\leq 2\,|\mathcal{S}| \exp\left(-\lfloor t/\tau \rfloor\right).
\end{aligned}$$

The last inequality follows from Note 3 and Note 4. ■

Recall that a reversible Markov chain has a transition probability matrix and stationary distribution satisfying the detailed balance equations

$$\pi_i p_{ij} = \pi_j p_{ji},$$

for all i, j.

Lemma 22. *Let $M = (\mathcal{S}, P)$ be a finite ergodic reversible Markov chain, and let π be its stationary distribution. Order the eigenvalues of P such that $1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \ldots$. Then*

$$\|D(t)\|_\lambda = |\lambda_2|^t.$$

Furthermore, if M has mixing time τ, we have that $|\lambda_2|^t \leq 2\exp\left(-\lfloor t/\tau \rfloor\right)$.

Proof. As P is reversible,

$$\sqrt{\frac{\pi_i}{\pi_j}} p_{ij}^{(t)} = \sqrt{\frac{\pi_i}{\pi_j}}\left(\frac{\pi_j}{\pi_i}\right) p_{ji}^{(t)} = \sqrt{\frac{\pi_j}{\pi_i}} p_{ji}^{(t)}, \tag{29}$$

and hence $D(t)' = D(t)$. Given a polynomial $\mathrm{f}(\cdot)$ and the symmetric matrix A, $Ax = \lambda x$ implies $\mathrm{f}(A)x = \mathrm{f}(\lambda)x$. Thus,

$$\|D(t)\|_\lambda = \sqrt{\lambda_{\max}\left(D(t)'D(t)\right)} = \sqrt{\lambda_{\max}\left(D(t)^2\right)} = \max_i \left|\lambda_i(D(t))\right|,$$

where $\lambda_1(D(t)), \lambda_2(D(t)), \ldots$ are the eigenvalues of $D(t)$. The matrix $D(t)$ is similar to $(P^t - e\pi')$ via the matrix $\Pi^{\frac{1}{2}}$, and hence has the same eigenvalues. Let $x_1, x_2, x_3, \ldots$ be the left eigenvectors of P, labelled with the indices of their associated eigenvalues. Then

$$x_i' \left(P^t - e\pi'\right) = x_i' \left(P^t - \lim_{n\to\infty} P^n\right) = \lambda_i^t x_i' - \lim_{n\to\infty} \lambda_i^n x_i' = \begin{cases} 0 & i = 1, \\ \lambda_i^t x_i' & i \neq 1. \end{cases}$$

Therefore λ_2^t is the greatest magnitude eigenvalue of $D(t)$. Furthermore, if $x \neq 0$ is a right eigenvector of $D(t)$ with eigenvalue λ, we have

$$\sum_{j\in\mathcal{S}} \sqrt{\frac{\pi_i}{\pi_j}} \left(p_{ij}^{(t)} - \pi_j\right) x_j = \frac{1}{\sqrt{\pi_i}} \sum_{j\in\mathcal{S}} \left(p_{ji}^{(t)} - \pi_i\right) \sqrt{\pi_j} x_j = \lambda x_i, \qquad \text{from (29)},$$

and so

$$\begin{aligned} |\lambda| \sum_{i\in\mathcal{S}} \sqrt{\pi_i}|x_i| &= \sum_{i\in\mathcal{S}} \left| \sum_{j\in\mathcal{S}} \left(p_{ji}^{(t)} - \pi_i\right) \sqrt{\pi_j} x_j \right| \\ &\leq \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}} \left|p_{ji}^{(t)} - \pi_i\right| \sqrt{\pi_j}|x_j| \\ &\leq \left(\max_{i\in\mathcal{S}} \sum_{k\in\mathcal{S}} \left|p_{ik}^{(t)} - \pi_k\right| \right) \sum_{j\in\mathcal{S}} \sqrt{\pi_j}|x_j| \\ &= 2 \max_{i\in\mathcal{S}} \mathrm{d}_{TV}\left(P_i^t, \pi\right) \sum_{j\in\mathcal{S}} \sqrt{\pi_j}|x_j|. \end{aligned}$$

So from Note 3 and Note 4 we have that $|\lambda| \leq 2\exp(-\lfloor t/\tau \rfloor)$. ■

Lemma 23. *Let $M = (S, P)$ be a finite ergodic Markov chain, and let π be its stationary distribution. Let $\{X_t\}$ be the process generated by M starting $X_0 \sim \pi$. For any two functions $\mathrm{f}, \mathrm{g} : \mathcal{S} \to \mathbb{R}$*

$$\left|\mathbb{E}\left[\left(\mathrm{f}(X_s) - \mathbb{E}_\pi \mathrm{f}(i)\right)\left(\mathrm{g}(X_{s+t}) - \mathbb{E}_\pi \mathrm{g}(i)\right)\right]\right| \leq \|D(t)\|_\lambda \sqrt{\mathbb{E}_\pi\left(\mathrm{f}(i) - \mathbb{E}_\pi \mathrm{f}(i)\right)^2 \mathbb{E}_\pi\left(\mathrm{g}(i) - \mathbb{E}_\pi \mathrm{g}(i)\right)^2}.$$

Proof. Denoting $\underline{\mathrm{f}}$ to be the column vector of $\mathrm{f}(x) - \mathbb{E}_\pi \mathrm{f}(i)$ over the states $x \in \mathcal{S}$, then

$$\begin{aligned} \left|\mathbb{E}\left[\underline{\mathrm{f}}_{X_s} \underline{\mathrm{g}}_{X_{s+t}}\right]\right| &= \left|\underline{\mathrm{f}}' \Pi P^t \underline{\mathrm{g}}\right| \\ &= \left|\underline{\mathrm{f}}' \Pi \left(P^t - e\pi'\right) \underline{\mathrm{g}}\right| \\ &= \left|\underline{\mathrm{f}}' \Pi^{\frac{1}{2}} D(t) \Pi^{\frac{1}{2}} \underline{\mathrm{g}}\right| \\ &\leq \left\|\underline{\mathrm{f}}' \Pi^{\frac{1}{2}}\right\|_2 \left\|D(t) \Pi^{\frac{1}{2}} \underline{\mathrm{g}}\right\|_2 \qquad \text{(Schwartz)} \\ &\leq \|D(t)\|_\lambda \left\|\underline{\mathrm{f}}' \Pi^{\frac{1}{2}}\right\|_2 \left\|\Pi^{\frac{1}{2}} \underline{\mathrm{g}}\right\|_2 \qquad \text{(Note 2)} \\ &= \|D(t)\|_\lambda \sqrt{\mathbb{E}_\pi\left(\underline{\mathrm{f}}_i\right)^2 \mathbb{E}_\pi\left(\underline{\mathrm{g}}_i\right)^2}. \end{aligned}$$

■

Lemma 23 shows how covariance terms can be bounded by the variance under the stationary distribution attenuated by the spectral norm of the covariance decay matrix. Combining this with Lemma 23 (or Lemma 22 for reversible chains) gives us Theorem 2.

Proof of Theorem 2. By the application of Lemma 21 and Lemma 23,

$$\begin{aligned} |\mathrm{Cov}_\pi(t;\mathrm{f})| &\leq \|D(t)\|_\lambda \mathrm{Var}_\pi(\mathrm{f}) \qquad \text{(Lemma 23)} \\ &\leq \sqrt{2|\mathcal{S}|\exp(-\lfloor t/\tau \rfloor)}\mathrm{Var}_\pi(\mathrm{f}) \qquad \text{(Lemma 21)} \\ &\leq \sqrt{2|\mathcal{S}|e}\sqrt{\exp(-t/\tau)}\mathrm{Var}_\pi(\mathrm{f}). \end{aligned}$$

This shows that Theorem 2 holds with some $\mathbf{L} \leq \sqrt{2|\mathcal{S}|e}$ and $0 \leq \alpha \leq \exp(-1/(2\tau))$. If the chain is reversible, then similarly, using Lemma 22, the bound of Theorem 2 holds with $\mathbf{L} = 2e$ and $\alpha = \exp(-1/\tau)$. ■

We can use the result of Theorem 2 to prove Theorem 3. Recall that Theorem 3 shows how the variance of an average of dependent samples can be bounded by $O(1/T)$ times the variance of a sample distributed according to the stationary distribution.

Proof of Theorem 3.

$$\begin{aligned} \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\right) &= \frac{1}{T^2}\mathbb{E}\left(\sum_{t=0}^{T-1}(\mathrm{f}(X_t)-\mathbb{E}\mathrm{f}(X_t))\right)^2 \\ &= \frac{1}{T^2}\sum_{t_1=0}^{T-1}\sum_{t_2=0}^{T-1}\mathbb{E}\left[(\mathrm{f}(X_{t_1})-\mathbb{E}\mathrm{f}(X_{t_1}))(\mathrm{f}(X_{t_2})-\mathbb{E}\mathrm{f}(X_{t_2}))\right] \\ &= \frac{1}{T^2}\sum_{t_1=0}^{T-1}\sum_{t_2=0}^{T-1}\mathrm{Cov}_\pi(|t_2-t_1|;\mathrm{f}) \\ &= \frac{1}{T^2}\sum_{t=-(T-1)}^{T-1}(T-|t|)\,\mathrm{Cov}_\pi(|t|;\mathrm{f}). \end{aligned}$$

Then, using Theorem 2,

$$\begin{aligned} \frac{1}{T^2}\sum_{t=-(T-1)}^{T-1}(T-|t|)\,\mathrm{Cov}_\pi(|t|;\mathrm{f}) &\leq \frac{1}{T^2}\sum_{t=-(T-1)}^{T-1}(T-|t|)\,\mathbf{L}\alpha^{|t|}\mathrm{Var}(\mathrm{f}(X)) \\ &= \frac{\mathbf{L}}{T^2}\left(\frac{T(1+\alpha)}{1-\alpha}-\frac{2\alpha(1-\alpha^T)}{(1-\alpha)^2}\right)\mathrm{Var}(\mathrm{f}(X)) \\ &\leq \frac{1}{T}\left(\frac{\mathbf{L}(1+\alpha)}{1-\alpha}\right)\mathrm{Var}(\mathrm{f}(X)), \end{aligned}$$

where the equality follows from

$$\sum_{t=-(T-1)}^{T-1}(T-|t|)\,\alpha^{|t|} = T + 2T\sum_{t=1}^{T-1}\alpha^t - 2\sum_{t=1}^{T-1}t\alpha^t$$

$$
\begin{aligned}
&= T + \frac{2T\alpha\left(1-\alpha^{T-1}\right)}{1-\alpha} - 2\sum_{s=1}^{T-1}\sum_{t=s}^{T-1}\alpha^t \\
&= \left(T + \frac{2T\alpha}{1-\alpha}\right) - \frac{2T\alpha^T}{1-\alpha} - 2\sum_{s=1}^{T-1}\sum_{t=s}^{T-1}\alpha^t \\
&= \frac{T\left(1+\alpha\right)}{1-\alpha} - \frac{2T\alpha^T}{1-\alpha} - 2\sum_{s=1}^{T-1}\alpha^s\frac{1-\alpha^{T-s}}{1-\alpha} \\
&= \frac{T\left(1+\alpha\right)}{1-\alpha} - \frac{2T\alpha^T}{1-\alpha} - \frac{2\alpha\left(1-\alpha^{T-1}\right)}{\left(1-\alpha\right)^2} + \frac{2\left(T-1\right)\alpha^T}{1-\alpha} \\
&= \frac{T\left(1+\alpha\right)}{1-\alpha} - \frac{2\alpha\left(1-\alpha^{T-1}\right) + 2\alpha^T\left(1-\alpha\right)}{\left(1-\alpha\right)^2} \\
&= \frac{T\left(1+\alpha\right)}{1-\alpha} - \frac{2\alpha\left(1-\alpha^{T}\right)}{\left(1-\alpha\right)^2}.
\end{aligned}
$$

We may set the Ω^* in Theorem 3 to $\Omega^* = \mathbf{L}\left(1+\alpha\right)/\left(1-\alpha\right)$. Furthermore, recalling that $\alpha \leq \exp(-1/(2\tau))$, we have

$$\Omega^* = \mathbf{L}\frac{1+\alpha}{1-\alpha} \leq 2\mathbf{L}\frac{1}{1-\exp\left(-1/(2\tau)\right)} \leq 6\mathbf{L}\tau,$$

where the last inequality uses $\left[1-\exp(-1/(2\tau))\right]^{-1} \leq \frac{8}{3}\tau$. Note that for $x = 1/(2\tau)$ we have $0 \leq x \leq 1/2$, and that for such an x

$$
\begin{aligned}
\exp(-x) &\leq 1 - x + \frac{x^2}{2} \\
\Leftrightarrow \quad 1-\exp(-x) &\geq x\left(1-\frac{x}{2}\right) \\
\Leftrightarrow \quad \frac{1}{1-\exp(-x)} &\leq \frac{1}{x}\cdot\frac{2}{2-x} \\
\Rightarrow \quad \frac{1}{1-\exp(-x)} &\leq \frac{4}{3x}.
\end{aligned}
\tag{30}
$$

■

Theorem 4 gives a result similar to Theorem 3, but without relying on Theorem 2, and hence without relying on the size of the state space. For the proof we find it useful to define the following.

Definition 9. *The* triangular discrimination *(Topsøe, 2000) between two distributions p,q on the finite set $\mathcal{S}$ is given by*

$$\mathrm{d}_{\triangle}(p,q) \stackrel{\text{def}}{=} \sum_{i\in\mathcal{S}}\frac{\left(p_i-q_i\right)^2}{p_i+q_i}.$$

Note 5. *We have that* $\mathrm{d}_{\triangle}(p,q) \leq 2\mathrm{d}_{TV}(p,q)$.

Note 5 can be seen from $\sum_{i\in\mathcal{S}}\frac{\left(p_i-q_i\right)^2}{p_i+q_i} = \sum_{i\in\mathcal{S}}\frac{|p_i-q_i|}{p_i+q_i}|p_i-q_i| \leq \sum_{i\in\mathcal{S}}|p_i-q_i|$.

Proof of Theorem 4. Write $g_i = \mathrm{f}(i) - \mathbb{E}\mathrm{f}(X)$, and write $V = \sum_{i\in\mathcal{S}} \pi_i g_i^2$, the variance of $\mathrm{f}(X)$. We have that $|g_i| \leq 2c$ for all $i \in \mathcal{S}$. Now, we have for any $s \geq 0$ and $t \geq 0$,

$$\begin{aligned}
&\mathbb{E}\left(\mathrm{f}(X_s) - \mathbb{E}\mathrm{f}(X_s)\right)\left(\mathrm{f}(X_{s+t}) - \mathbb{E}\mathrm{f}(X_{s+t})\right) \\
&= \sum_{i,j\in\mathcal{S}} \pi_i g_i \left(p_{ij}^{(t)} - \pi_j\right) g_j \\
&= \sum_{i,j\in\mathcal{S}} \sqrt{\pi_i} \frac{p_{ij}^{(t)} - \pi_j}{\sqrt{p_{ij}^{(t)} + \pi_j}} \sqrt{\pi_i} g_i \sqrt{p_{ij}^{(t)} + \pi_j}\, g_j \\
&\leq \left(\sum_{i\in\mathcal{S}} \pi_i \sum_{j\in\mathcal{S}} \frac{\left(p_{ij}^{(t)} - \pi_j\right)^2}{p_{ij}^{(t)} + \pi_j}\right)^{1/2} \left(\sum_{i\in\mathcal{S}} \pi_i g_i^2 \sum_{j\in\mathcal{S}} \left(p_{ij}^{(t)} + \pi_j\right) g_j^2\right)^{1/2} \qquad \text{(Schwartz)} \\
&= \left(\sum_{i\in\mathcal{S}} \pi_i \mathrm{d}_{\triangle}\left(P_i^t, \pi\right)\right)^{1/2} \left(2V^2 + \sum_{i\in\mathcal{S}} \pi_i g_i^2 \sum_{j\in\mathcal{S}} \left(p_{ij}^{(t)} - \pi_j\right) g_j^2\right)^{1/2} \\
&\leq (2d_t)^{1/2} \left(2V^2 + (2c)^2 \sum_{i\in\mathcal{S}} \pi_i g_i^2 \sum_{j\in\mathcal{S}} \left|p_{ij}^{(t)} - \pi_j\right|\right)^{1/2} \qquad \text{(Note 5)} \\
&\leq 2d_t^{1/2} \left(V^2 + 2c^2 V d_t\right)^{1/2}. \qquad (31)
\end{aligned}$$

Consider the case where $V = \mathrm{Var}\left(\mathrm{f}(X)\right) > \varepsilon$. If $d_t \leq \varepsilon$, from Equation (31), we have

$$\mathbb{E}\left(\mathrm{f}(X_s) - \mathbb{E}\mathrm{f}(X_s)\right)\left(\mathrm{f}(X_{s+t}) - \mathbb{E}\mathrm{f}(X_{s+t})\right) \leq 2\sqrt{2}(1+c) d_t^{1/2} V. \qquad (32)$$

This holds for all s,t such that $d_t \leq \varepsilon$, which is implied by

$$\begin{aligned}
& \exp\left(-t/\tau + 1\right) \leq \varepsilon \\
\Leftrightarrow \quad & -t/\tau \leq \ln\varepsilon - 1 \\
\Leftrightarrow \quad & t \geq \tau\left(1 + \ln\frac{1}{\varepsilon}\right) \\
\Leftarrow \quad & t \geq 2\tau \ln\frac{1}{\varepsilon},
\end{aligned}$$

as $\varepsilon \leq e^{-1}$. For all s,t we have

$$\mathbb{E}\left(\mathrm{f}(X_s) - \mathbb{E}\mathrm{f}(X_s)\right)\left(\mathrm{f}(X_{s+t}) - \mathbb{E}\mathrm{f}(X_{s+t})\right) \leq V, \qquad (33)$$

which is a Cauchy-Schwartz inequality:

$$\begin{aligned}
&\mathbb{E}\left(\mathrm{f}(X_s) - \mathbb{E}\mathrm{f}(X_s)\right)\left(\mathrm{f}(X_{s+t}) - \mathbb{E}\mathrm{f}(X_{s+t})\right) \\
&= \sum_{i,j\in\mathcal{S}} \pi_i g_i p_{ij}^{(t)} g_j \\
&= \sum_{i,j\in\mathcal{S}} \sqrt{\pi_i p_{ij}^{(t)}}\, g_i \sqrt{\pi_i p_{ij}^{(t)}}\, g_j \\
&\leq \left(\sum_{i\in\mathcal{S}} \pi_i g_i^2 \sum_{j\in\mathcal{S}} p_{ij}^{(t)}\right)^{1/2} \left(\sum_{j\in\mathcal{S}} \left(\sum_{i\in\mathcal{S}} \pi_i p_{ij}^{(t)}\right) g_j^2\right)^{1/2}
\end{aligned}$$

$$
\begin{aligned}
&= \left(\sum_{i\in\mathcal{S}} \pi_i g_i^2\right)^{1/2} \left(\sum_{j\in\mathcal{S}} \pi_j g_j^2\right)^{1/2} \\
&= V.
\end{aligned}
$$

So from Equation (32) and Equation (33) we have

$$
\begin{aligned}
&\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} \mathrm{f}(X_t)\right) \\
&= \frac{1}{T^2}\sum_{t_1=0}^{T-1}\sum_{t_2=0}^{T-1} \mathbb{E}\left(\mathrm{f}(X_{t_1}) - \mathbb{E}\mathrm{f}(X_{t_1})\right)\left(\mathrm{f}(X_{t_2}) - \mathbb{E}\mathrm{f}(X_{t_2})\right) \\
&= \frac{1}{T^2}\sum_{t=0}^{T-1} \mathbb{E}\left(\mathrm{f}(X_t) - \mathbb{E}\mathrm{f}(X_t)\right)^2 + \frac{2}{T^2}\sum_{s=0}^{T-2}\sum_{t=1}^{T-s-1} \mathbb{E}\left(\mathrm{f}(X_s) - \mathbb{E}\mathrm{f}(X_s)\right)\left(\mathrm{f}(X_{s+t}) - \mathbb{E}\mathrm{f}(X_{s+t})\right) \\
&= \frac{1}{T^2}\sum_{t=0}^{T-1} \mathbb{E}\left(\mathrm{f}(X) - \mathbb{E}\mathrm{f}(X)\right)^2 + \frac{2}{T^2}\sum_{t=1}^{T-1} (T-t)\mathbb{E}\left(\mathrm{f}(X_0) - \mathbb{E}\mathrm{f}(X_0)\right)\left(\mathrm{f}(X_t) - \mathbb{E}\mathrm{f}(X_t)\right) \\
&= \frac{1}{T}V + \frac{2}{T^2}\sum_{t=1}^{\lfloor 2\tau\ln(1/\varepsilon)\rfloor} (T-t)\mathbb{E}\left(\mathrm{f}(X_0) - \mathbb{E}\mathrm{f}(X_0)\right)\left(\mathrm{f}(X_t) - \mathbb{E}\mathrm{f}(X_t)\right) \\
&\qquad + \frac{2}{T^2}\sum_{t=\lfloor 2\tau\ln(1/\varepsilon)\rfloor+1}^{T-1} (T-t)\mathbb{E}\left(\mathrm{f}(X_0) - \mathbb{E}\mathrm{f}(X_0)\right)\left(\mathrm{f}(X_t) - \mathbb{E}\mathrm{f}(X_t)\right) \\
&\leq \frac{1}{T}V + 4\tau\ln\left(\varepsilon^{-1}\right)\frac{1}{T}V + 4\sqrt{2}(1+c)\sum_{t=\lfloor 2\tau\ln(1/\varepsilon)\rfloor+1}^{\infty} d_t^{1/2}\,\frac{1}{T}V \\
&\leq \left(1 + 4\tau\ln\frac{1}{\varepsilon} + 25\tau(1+c)\varepsilon\right)\frac{1}{T}V, \qquad (34)
\end{aligned}
$$

where the last line follows from

$$
\begin{aligned}
\sum_{t=\lfloor 2\tau\ln(1/\varepsilon)\rfloor+1}^{\infty} d_t^{1/2} &\leq \sum_{t=\lfloor 2\tau\ln(1/\varepsilon)\rfloor+1}^{\infty} \exp\left(-t/(2\tau) + 1/2\right) \\
&= \sqrt{e}\sum_{t=\lfloor 2\tau\ln(1/\varepsilon)\rfloor+1}^{\infty} \exp\left(-t/(2\tau)\right) \\
&\leq \sqrt{e}\exp\left(-\ln\frac{1}{\varepsilon}\right)\sum_{t=0}^{\infty} \left(\exp\left(-1/(2\tau)\right)\right)^t \\
&= \sqrt{e}\varepsilon\frac{1}{1-\exp(-1/(2\tau))} \\
&\leq \frac{8\sqrt{e}}{3}\tau\varepsilon,
\end{aligned}
$$

where we have again used Equation (30). For the case where $\mathrm{Var}\left(\mathrm{f}(X)\right) \leq \varepsilon$ we have

$$
\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1} \mathrm{f}(X_t)\right) = \frac{1}{T^2}\sum_{t_1=0}^{T-1}\sum_{t_2=0}^{T-1} \mathbb{E}\left(\mathrm{f}(X_{t_1}) - \mathbb{E}\mathrm{f}(X_{t_1})\right)\left(\mathrm{f}(X_{t_2}) - \mathbb{E}\mathrm{f}(X_{t_2})\right)
$$

$$\begin{aligned} &\leq \frac{1}{T^2}\sum_{t_1=0}^{T-1}\sum_{t_2=0}^{T-1} V \\ &\leq \varepsilon. \end{aligned} \tag{35}$$

As the variance is bounded either by Equation (34) or by Equation (35), taking their sum gives the result. ■

Lastly, we prove the corollary to Theorem 4, which shows the essential rate of decrease of the bound.

Proof of Corollary 5. Selecting ε such that, writing $V = \mathrm{Var}(\mathrm{f}(X))$,

$$\frac{1}{\varepsilon} = \frac{T}{4\tau V} + \frac{25}{4}(1+c),$$

satisfies $0 \leq \varepsilon \leq e^{-1}$. Substituting this into the result of Theorem 4 gives

$$\begin{aligned} \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\right) &\leq \frac{4\tau V}{T+25(1+c)\tau V} + \left(1 + \frac{100(1+c)\tau^2 V}{T+25(1+c)\tau V}\right. \\ &\qquad \left. + 4\tau\ln\left(\frac{T}{4\tau V} + \frac{25}{4}(1+c)\right)\right)\frac{V}{T} \\ &\leq \frac{4\tau V}{T} + \left(1 + 4\tau + 4\tau\ln\left(\frac{T}{4\tau V} + \frac{25}{4}(1+c)\right)\right)\frac{V}{T} \\ &\leq (1+8\tau)\frac{V}{T} + 4\tau\ln\left(7(1+c) + \frac{1}{4\tau}\left(\frac{V}{T}\right)^{-1}\right)\frac{V}{T}. \end{aligned}$$

■

Appendix C. Proofs for Section 5.1

In this section we give the proofs for Lemma 6 and Theorem 7 in Section 5.1. A few auxiliary lemmas are also given.

Proof of Lemma 6. Consider $\mathcal{F}$-measurable random variables A, B, with $\mathcal{F}$ being some σ-algebra. If B is also $\mathcal{G}$-measurable for some $\mathcal{G} \subset \mathcal{F}$ such that $\mathbb{E}[A|\mathcal{G}] = B$ almost surely, then we have:

$$\mathbb{E}[A-B] = 0; \qquad \text{and} \qquad \mathbb{E}[B(A-B)] = 0$$

(Note that $\mathbb{E}[B(A-B)|\mathcal{G}] = B\mathbb{E}[A-B|\mathcal{G}] = 0$, almost surely). This gives us

$$\begin{aligned} \mathrm{Var}(A) &= \mathbb{E}\left[(A-\mathbb{E}[A])^2\right] \\ &= \mathbb{E}\left[((B-\mathbb{E}[B]) + (A-B) - \mathbb{E}[A-B])^2\right] \\ &= \mathbb{E}\left[((B-\mathbb{E}[B]) + (A-B))^2\right] \\ &= \mathbb{E}\left[(B-\mathbb{E}[B])^2 + 2(B-\mathbb{E}[B])(A-B) + (A-B)^2\right] \\ &= \mathbb{E}\left[(B-\mathbb{E}[B])^2\right] + 2\mathbb{E}[B(A-B)] - 2\mathbb{E}[B]\mathbb{E}[A-B] + \mathbb{E}\left[(A-B)^2\right] \\ &= \mathrm{Var}(B) + \mathbb{E}\left[(A-B)^2\right]. \end{aligned} \tag{36}$$

Now choosing $\mathcal{F}$ to be the smallest σ-algebra such that the random variable $(X_0,\ldots,X_{T-1},J_0,\ldots,J_{T-1})$ and our functions $\mathrm{f},\mathrm{J},\mathrm{a}$, for all X_t, are measurable, and $\mathcal{G}$ such that $(X_0,\ldots,X_{T-1})$, and the functions on X_t, are measurable, we have that for

$$A=\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(J_t-\mathrm{a}(X_t)\right)\qquad\text{and}\qquad B=\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(\mathrm{J}(X_t)-\mathrm{a}(X_t)\right),$$

A and B are $\mathcal{F}$-measurable, B is $\mathcal{G}$-measurable, and $\mathcal{G}\subset\mathcal{F}$. Furthermore, we have

$$\begin{aligned}
\mathbb{E}[A|\,\mathcal{G}] &= \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(J_t-\mathrm{a}(X_t)\right)\middle|X_0,\ldots,X_{T-1}\right]\\
&= \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathrm{f}(X_t)\left(J_t-\mathrm{a}(X_t)\right)|X_t\right]\\
&= \frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(\mathbb{E}\left[J_t|X_t\right]-\mathrm{a}(X_t)\right)\\
&= \frac{1}{T}\sum_{t=0}^{T-1}\mathrm{f}(X_t)\left(\mathrm{J}(X_t)-\mathrm{a}(X_t)\right)\\
&= B.
\end{aligned}$$

The proof then follows from Equation 36. ■

The proof of Theorem 7 requires some additional tools. In addition to (10) we also consider a variation of GPOMDP where a fixed length chain is used to estimate the discounted value function:

$$\Delta_T^{(S)}\stackrel{\text{def}}{=}\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}J_{t+1}^{(S)},\qquad J_t^{(S)}\stackrel{\text{def}}{=}\sum_{s=t}^{t+S-1}\beta^{s-t}\mathrm{r}(X_s).$$

Lemma 24. *Let $D=(\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,\mathrm{r},\mu)$ be a controlled POMDP satisfying Assumptions 1, 2 and 3. Then*

$$\left\|\Delta_T^{(+S)}-\Delta_T^{(S)}\right\|\leq\frac{\mathbf{BR}}{1-\beta}\beta^S,$$

and similarly,

$$\left\|\Delta_T^{(\infty)}-\Delta_T^{(S)}\right\|\leq\frac{\mathbf{BR}}{1-\beta}\beta^S,$$

where $\Delta_T^{(\infty)}$ denotes $\Delta_T^{(S)}$ in the limit as $S\to\infty$.

Proof.

$$\begin{aligned}
\left\|\Delta_T^{(+S)}-\Delta_T^{(S)}\right\| &= \left\|\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}J_{t+1}^{(+S)}-\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}J_{t+1}^{(S)}\right\|\\
&= \left\|\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\sum_{s=t+1+S}^{c}\beta^{s-t-1}\mathrm{r}(X_s)\right\|,\qquad c=T+S
\end{aligned}$$

$$\begin{aligned}
&\leq \mathbf{BR}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{s=t+1+S}^{c}\beta^{s-t-1}\\
&= \mathbf{BR}\frac{1}{T}\sum_{t=0}^{T-1}\frac{\beta^S(1-\beta^{c-S-t-1})}{1-\beta}\\
&\leq \frac{\mathbf{BR}}{1-\beta}\beta^S.
\end{aligned}$$

Obtain the bound $\left\|\Delta_T^{(\infty)}-\Delta_T^{(S)}\right\|$ similarly by considering the limit as $c\to\infty$. ■

Lemma 25. *Let $D=(\mathcal{S},\mathcal{U},\mathcal{Y},P,\nu,r,\mu)$ be a controlled POMDP satisfying Assumptions 1, 2 and 3. Let $\{Z_t\}=\{X_t,Y_t,U_t,X_{t+1}\}$ be the process generated by D. For any $a:\mathcal{S}\times\mathcal{Y}\times\mathcal{U}\times\mathcal{S}\to\mathbb{R}$ satisfying $|a(\cdot)|\leq\mathbf{M}$, we have*

$$\begin{aligned}
\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)}-a(Z_t)\right)\right) &\leq \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-a(Z_t)\right)\right)\\
&\quad+\frac{5\mathbf{B}^2\mathbf{R}(\mathbf{R}+\mathbf{M})}{(1-\beta)^2}\beta^S
\end{aligned}$$

and similarly,

$$\begin{aligned}
\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-a(Z_t)\right)\right) &\leq \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(\infty)}-a(Z_t)\right)\right)\\
&\quad+\frac{5\mathbf{B}^2\mathbf{R}(\mathbf{R}+\mathbf{M})}{(1-\beta)^2}\beta^S,
\end{aligned}$$

where $J_t^{(\infty)}$ denotes $J_t^{(S)}$ in the limit as $S\to\infty$.

Proof.

$$\begin{aligned}
&\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)}-a(Z_t)\right)\right)\\
&= \mathbb{E}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)}-a(Z_t)\right)-\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)}-a(Z_t)\right)\right]\right)^2\\
&= \mathbb{E}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-a(Z_t)\right)-\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-a(Z_t)\right)\right]\right.\\
&\qquad\left.+\left(\Delta_T^{(+S)}-\Delta_T^{(S)}\right)-\mathbb{E}\left[\Delta_T^{(+S)}-\Delta_T^{(S)}\right]\right)^2\\
&= \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-a(Z_t)\right)\right)+\mathbb{E}\left(\Delta_T^{(+S)}-\Delta_T^{(S)}\right)^2-\left(\mathbb{E}\left[\Delta_T^{(+S)}-\Delta_T^{(S)}\right]\right)^2\\
&\quad+2\mathbb{E}\left[\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-a(Z_t)\right)-\mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-a(Z_t)\right)\right]\right)'\right.\\
&\qquad\left.\times\left(\Delta_T^{(+S)}-\Delta_T^{(S)}\right)\right]
\end{aligned}$$

$$
\begin{aligned}
&\leq \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right)+\mathbb{E}\left\|\Delta_T^{(+S)}-\Delta_T^{(S)}\right\|^2 \\
&\quad+2\mathbb{E}\left[\left(\left\|\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right\|+\mathbb{E}\left\|\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right\|\right)\right. \\
&\qquad\left.\times\left\|\Delta_T^{(+S)}-\Delta_T^{(S)}\right\|\right] \\
&\leq \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right)+\left(\frac{\mathbf{BR}}{1-\beta}\beta^S\right)^2 \\
&\quad+4\mathbb{E}\left\|\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right\|\frac{\mathbf{BR}}{1-\beta}\beta^S \qquad \text{(Lemma 24)} \\
&\leq \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right)+\left(\frac{\mathbf{BR}}{1-\beta}\beta^S\right)^2 \\
&\quad+4\left(\frac{\mathbf{B}(\mathbf{R}+\mathbf{M}(1-\beta))}{1-\beta}\right)\left(\frac{\mathbf{BR}}{1-\beta}\beta^S\right) \\
&\leq \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right)+\frac{5\mathbf{B}^2\mathbf{R}(\mathbf{R}+\mathbf{M})}{(1-\beta)^2}\beta^S.
\end{aligned}
$$

Obtain the second result by replacing $J_t^{(+S)}$ with $J_t^{(S)}$, and $J_t^{(S)}$ with $J_t^{(\infty)}$; then $\Delta_T^{(+S)}-\Delta_T^{(S)}$ becomes $\Delta_T^{(S)}-\Delta_T^{(\infty)}$. ■

Using these Lemmas, and Theorem 4, we can now prove Theorem 7.

Proof of Theorem 7. In this proof we will apply Theorem 4 to show that the variance of the sample average is $O(\ln(T)/T)$ times the variance of a single sample, and we will apply Lemma 6 to show that the additional variance due to estimating the value function need not be considered. We first use Lemma 25 to convert each of the samples within the average to be functions on a fixed length of the chain, that is, functions on states of the Markov process $\{X_t, Y_t, U_t, \ldots, U_{t+S-1}, X_{t+S}\}$. We can then use Theorem 4 for the sample average of functions on this process. Write

$$V = \mathrm{Var}_\pi\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\left(\mathrm{J}_\beta(j)-\mathrm{a}(i,y,u,j)\right)\right),$$

$$E = \mathbb{E}_\pi\left[\left(\frac{\nabla\mu_u(y)}{\mu_u(y)}\left(g(j)-\mathrm{J}_\beta(j)\right)\right)^2\right],$$

and

$$C = \frac{5\mathbf{B}^2\mathbf{R}(\mathbf{R}+\mathbf{M})}{(1-\beta)^2}\beta^S,$$

note that

$$1+\left\|\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right\|_\infty \leq \frac{1}{7}\cdot\frac{C_1}{1-\beta},$$

where $\|a\|_\infty$ is the maximum of the magnitudes of the components of vector a, and denote the mixing time of the process $\{X_t, Y_t, U_t, \ldots, U_{t+S-1}, X_{t+S}\}$ by $\tilde{\tau}$. We have

$$\begin{aligned}
&\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)}-\mathrm{a}(Z_t)\right)\right)\\
&\leq\ \mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(S)}-\mathrm{a}(Z_t)\right)\right)+C \qquad \text{(Lemma 25)}\\
&\leq\ K\varepsilon+\left(1+\frac{25}{7}\tilde{\tau}\frac{C_1}{1-\beta}\varepsilon+4\tilde{\tau}\ln\frac{1}{\varepsilon}\right)\frac{1}{T}\mathrm{Var}\left(\frac{\nabla\mu_{U_0}(Y_0)}{\mu_{U_0}(Y_0)}\left(J_1^{(S)}-\mathrm{a}(Z_0)\right)\right)\\
&\qquad +C \qquad \text{(Theorem 4)}\\
&\leq\ K\varepsilon+\left(1+\frac{25}{7}\tilde{\tau}\frac{C_1}{1-\beta}\varepsilon+4\tilde{\tau}\ln\frac{1}{\varepsilon}\right)\frac{1}{T}\mathrm{Var}\left(\frac{\nabla\mu_{U_0}(Y_0)}{\mu_{U_0}(Y_0)}\left(J_1^{(\infty)}-\mathrm{a}(Z_0)\right)\right)\\
&\qquad +\left(1+\frac{25}{7}\tilde{\tau}\frac{C_1}{1-\beta}\varepsilon+4\tilde{\tau}\ln\frac{1}{\varepsilon}\right)\frac{C}{T}+C \qquad \text{(Lemma 25)}\\
&=\ K\varepsilon+\left(1+\frac{25}{7}\tilde{\tau}\frac{C_1}{1-\beta}\varepsilon+4\tilde{\tau}\ln\frac{1}{\varepsilon}\right)\left(\frac{V}{T}+\frac{E}{T}+\frac{C}{T}\right)+C \qquad \text{(Lemma 6).}
\end{aligned}$$

Here, Theorem 4 was applied to each of the K dimensions of the quantity the variance is taken over (recall that we consider the variance of a vector quantity to be the sum of the variance of its components). Now, similar to the proof of Corollary 5, we choose

$$\frac{1}{\varepsilon}=\frac{K}{4\tilde{\tau}}\left(\frac{V}{T}+\frac{E}{T}+\frac{C}{T}\right)^{-1}+\frac{25}{28}\cdot\frac{C_1}{1-\beta},$$

giving

$$\begin{aligned}
&\mathrm{Var}\left(\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\mu_{U_t}(Y_t)}{\mu_{U_t}(Y_t)}\left(J_{t+1}^{(+S)}-\mathrm{a}(Z_t)\right)\right)\\
&\leq\ K\varepsilon+\left(1+\frac{25}{7}\tilde{\tau}\frac{C_1}{1-\beta}\varepsilon+4\tilde{\tau}\ln\frac{1}{\varepsilon}\right)\left(\frac{V}{T}+\frac{E}{T}+\frac{C}{T}\right)+C\\
&\leq\ 4\tilde{\tau}\left(\frac{V}{T}+\frac{E}{T}+\frac{C}{T}\right)+\Bigg[1+4\tilde{\tau}\\
&\qquad +4\tilde{\tau}\ln\left(\frac{25}{28}\cdot\frac{C_1}{1-\beta}+\frac{K}{4\tilde{\tau}}\left(\frac{V}{T}+\frac{E}{T}+\frac{C}{T}\right)^{-1}\right)\Bigg]\left(\frac{V}{T}+\frac{E}{T}+\frac{C}{T}\right)+C\\
&\leq\ \mathrm{h}\left(\frac{\tilde{\tau}}{T}V\right)+\mathrm{h}\left(\frac{\tilde{\tau}}{T}E\right)+\mathrm{h}\left(\frac{\tilde{\tau}}{T}C\right)+C\\
&\leq\ \mathrm{h}\left(\frac{\tau\ln(e(S+1))}{T}V\right)+\mathrm{h}\left(\frac{\tau\ln(e(S+1))}{T}E\right)+\mathrm{h}\left(\frac{\tau\ln(e(S+1))}{T}C\right)+C,
\end{aligned}$$

where the last line follows from $\tilde{\tau}\leq\tau\ln(e(S+1))$ (Lemma 1), and from h being an increasing function. Lastly, we have

$$\mathrm{h}\left(\frac{\tau\ln(e(S+1))}{T}C\right)+C$$

$$
\begin{aligned}
&\leq \left(\frac{1}{T}+8\tau\frac{\ln e(S+1)}{T}\right.\\
&\quad \left.+4\tau\frac{\ln(e(S+1))}{T}\ln\left(\frac{C_1}{1-\beta}+\frac{K(1-\beta)^2}{20\tau\mathbf{B}^2\mathbf{R}(\mathbf{R}+\mathbf{M})\ln(e(S+1))}\left(\frac{\beta^S}{T}\right)^{-1}\right)+1\right)C\\
&\leq \left(\frac{1}{T}+8\tau\frac{\ln e(S+1)}{T}+4\tau\frac{\ln(T)\ln(e(S+1))}{T}\right.\\
&\quad \left.+4\tau\frac{S\ln(e(S+1))}{T}\ln\frac{1}{\beta}+4\tau\frac{\ln(e(S+1))}{T}\ln\left(\frac{C_1}{1-\beta}+\frac{K(1-\beta)^2}{20\tau\mathbf{B}^2\mathbf{R}(\mathbf{R}+\mathbf{M})}\right)+1\right)C\\
&\leq \frac{2C_2}{(1-\beta)^2}\left[\ln\frac{1}{\beta}+\ln\left(\frac{C_1}{1-\beta}+\frac{K(1-\beta)^2}{C_2}\right)\right]\frac{(T+S)\ln(e(S+1))}{T}\beta^S.
\end{aligned}
$$

The second step has used the increasing property of ln, along with $\ln(e(S+1)) \geq 1$ and $\beta^S/T \leq 1$. This gives us, for any $A, B \geq 0$,

$$
\ln\left(A+\frac{B}{\ln(e(S+1))}\left(\frac{\beta^S}{T}\right)^{-1}\right) \leq \ln\left(A+B\left(\frac{\beta^S}{T}\right)^{-1}\right) \leq \ln\left((A+B)\left(\frac{\beta^S}{T}\right)^{-1}\right).
$$

■

Appendix D. Value Function Example

Here we consider a somewhat less trivial example than that presented in Section 6.2—an example of reducing variance through appropriate choice of value function. A toy MDP is shown in Figure 7. Here action a_1 causes the MDP to have a tendency to stay in state s_1, and action a_2 causes the MDP to have a tendency to move away from s_1 and stay in state s_2 and s_3.

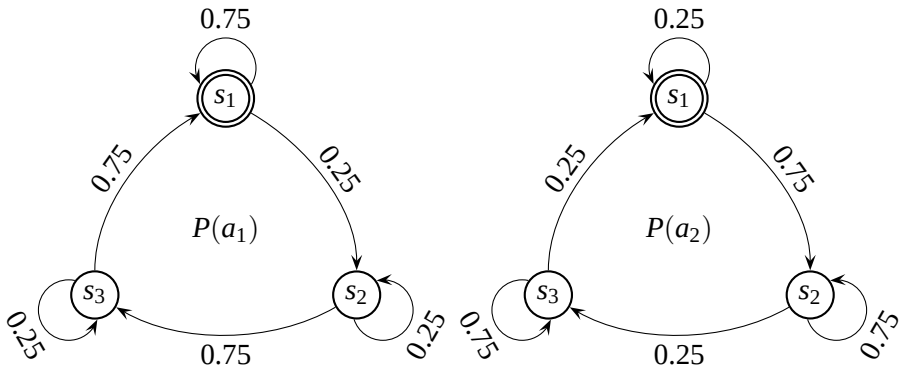


Figure 7: Transition probabilities for a toy 3 state, 2 action Markov decision process

Now consider the resultant controlled MDP when the single parameter, state independent policy

$$
\mu_{a_1} = \frac{e^\theta}{e^\theta + e^{-\theta}} \qquad \mu_{a_2} = 1 - \mu_{a_1} = \frac{e^{-\theta}}{e^\theta + e^{-\theta}}
$$

along with any reward function satisfying Assumption 2 is used. Note that this controlled MDP satisfies Assumptions 1, 2 and 3 for all θ. For the policy at $\theta = 0$ we have $\mu_{a_1} = \mu_{a_2} = 0.5$ and

$$
\nabla\mu_{a_1} = 0.5 \qquad \nabla\mu_{a_2} = -0.5.
$$

The transition matrix and stationary distribution of the resultant chain are:

$$P = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \end{bmatrix} \qquad \pi = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}.$$

In this case the 1×3 matrix $G = (1/6, -1/6, 0)$, and the right null space of G is $\{\alpha_1 v_1 + \alpha_2 v_2 : \alpha_1, \alpha_2 \in \mathbb{R}\}$, where

$$v_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \qquad v_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Any value function of the form $\mathrm{V} = \mathrm{J}_\beta + \alpha_1 v_1 + \alpha_2 v_2$ will produce an unbiased estimate of $\nabla_\beta \eta$. In this case we have that, writing $r_i = \mathrm{r}(s_i)$,

$$\mathrm{J}_\beta = (I - \beta P)^{-1} \mathrm{r} = \frac{2}{(2-\beta)^3 - \beta^3} \begin{bmatrix} (2-\beta)^2 & \beta(2-\beta) & \beta^2 \\ \beta^2 & (2-\beta)^2 & \beta(2-\beta) \\ \beta(2-\beta) & \beta^2 & (2-\beta)^2 \end{bmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}.$$

If we select $\beta = 0.9$ this becomes

$$\mathrm{J}_{0.9} = \frac{1}{0.301} \begin{bmatrix} 1.21 & 0.99 & 0.81 \\ 0.81 & 1.21 & 0.99 \\ 0.99 & 0.81 & 1.21 \end{bmatrix} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} = \frac{1}{0.301} \begin{pmatrix} 1.21 r_1 + 0.99 r_2 + 0.81 r_3 \\ 0.81 r_1 + 1.21 r_2 + 0.99 r_3 \\ 0.99 r_1 + 0.81 r_2 + 1.21 r_3 \end{pmatrix}.$$

If we had $\mathrm{r} = (1/10, 2/11, 0)'$ then we would again have $\mathrm{J}_{0.9} [= (1, 1, 81/99)']$ in the right null space of G, and we could again choose $\mathrm{V} = 0$ to obtain a zero bias, zero variance estimate of $\nabla_\beta \eta$. Consider instead the reward function

$$\mathrm{r}(i) = \begin{cases} 4.515 & i = s_1 \\ 0 & \text{otherwise,} \end{cases}$$

so that $\mathrm{J}_{0.9} = (18.15, 12.15, 14.85)'$ and $\nabla_{0.9} \eta = 1$. We now have

$$\begin{aligned} \mathrm{Var}_\pi \left(\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{J}_{0.9}(j) \right) &= \mathbb{E}_\pi \left(\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{J}_{0.9}(j) \right)^2 - \left(\mathbb{E}_\pi \left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{J}_{0.9}(j) \right] \right)^2 \\ &= \mathbb{E}_\pi \left(\mathrm{J}_{0.9}(j) \right)^2 - 1 \\ &= \pi' \begin{pmatrix} 18.15^2 \\ 12.15^2 \\ 14.85^2 \end{pmatrix} - 1 \\ &= 231.5225. \end{aligned}$$

The second line is obtained from $|\nabla \mu_u(i) / \mu_u(i)| = 1$ and $\nabla_{0.9} \eta = 1$. If we choose $\alpha_1 = -15.15\sqrt{2}$ and $\alpha_2 = -14.85$ then, for the value function $\mathrm{V} = \mathrm{J}_\beta + \alpha_1 v_1 + \alpha_2 v_2$, we have

$$\begin{aligned} \mathrm{Var}_\pi \left(\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j) \right) &= \mathbb{E}_\pi \left(\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j) \right)^2 - \left(\mathbb{E}_\pi \left[\frac{\nabla \mu_u(i)}{\mu_u(i)} \mathrm{V}(j) \right] \right)^2 \\ &= \pi' \begin{pmatrix} (18.15 - 15.15)^2 \\ (12.15 - 15.15)^2 \\ 0 \end{pmatrix} - 1 \\ &= 5; \end{aligned}$$

a significant reduction in variance, with no additional bias.

References

D. Aberdeen. A survey of approximate methods for solving partially observable markov decision processes. Technical report, Research School of Information Science and Engineering, Australian National University, Australia, 2002.

L. Baird. Gradient descent for general reinforcement learning. In S. A. Solla M. S Kearns and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. The MIT Press, 1999.

P. L. Bartlett and J. Baxter. Estimation and approximation bounds for gradient-based reinforcement learning. *Journal of Computer and System Sciences*, 64(1):133–150, February 2002.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846, 1983.

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

J. Baxter, P. L. Bartlett, and L. Weaver. Experiments with infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:351–381, 2001.

S. J. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 262:33–57, 1996.

P. Dayan. Reinforcement comparison. In *Proceedings of the 1990 Connectionist Models Summer School*, pages 45–51. Morgan Kaufmann, 1990.

J. L. Doob. *Measure Theory*. Number 143 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1994.

M. Evans and T. Swartz. *Approximating integrals via Monte Carlo and deterministic methods*. Oxford statistical science series. Oxford University Press, Oxford; New York, 2000.

G. S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer series in operations research. Springer-Verlag, New York, 1996.

P. W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

P. W. Glynn and P. L'Ecuyer. Likelihood ratio gradient estimation for regenerative stochastic recursions. *Advances in Applied Probability*, 12(4):1019–1053, 1995.

G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford University Press, Oxford, 1992.

J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Chapman and Hall, New York, 1965.

T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 345–352. The MIT Press, 1995.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

H. Kimura and S. Kobayashi. An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value functions. In *International Conference on Machine Learning*, pages 278–286, 1998a.

H. Kimura and S. Kobayashi. Reinforcement learning for continuous action using stochastic gradient ascent. In *Intelligent Autonomous Systems*, volume 5, pages 288–295, 1998b.

H. Kimura, K. Miyazaki, and S. Kobayashi. Reinforcement learning in POMDPs with function approximation. In D. H. Fisher, editor, *International Conference on Machine Learning*, pages 152–160, 1997.

H. Kimura, M. Yamamura, and S. Kobayashi. Reinforcement learning by stochastic hill climbing on discounted reward. In *International Conference on Machine Learning*, pages 295–303, 1995.

V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In T. K. Leen S. A. Solla and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. The MIT Press, 2000.

V. R. Konda and J. N. Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

W. S. Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, February 2001.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley series in probability and mathematical statistics. Applied probability and statistics. John Wiley & Sons, New York, 1994.

M. I. Reiman and A. Weiss. Sensitivity analysis for simulations via likelihood ratios. *Operations Research*, 37, 1989.

R. Y. Rubinstein. How to optimize complex stochastic systems from a single sample path by the score function method. *Annals of Operations Research*, 27:175–211, 1991.

E. Seneta. *Non-negative Matrices and Markov Chains*. Springer series in statistics. Springer-Verlag, New York, 1981.

S. P. Singh, T. Jaakkola, and M. I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *International Conference on Machine Learning*, pages 284–292, 1994.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge MA, 1998.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In T. K. Leen S. A. Solla and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. The MIT Press, 2000.

F. Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory*, 46:1602–1609, 2000.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Journal of Machine Learning Research 5 (2004) 1531–1555 Submitted 11/03; Revised 8/04; Published 11/04

Fast Binary Feature Selection with Conditional Mutual Information

François Fleuret FRANCOIS.FLEURET@EPFL.CH
EPFL – CVLAB
Station 14
CH-1015 Lausanne
Switzerland

Editor: Isabelle Guyon

Abstract

We propose in this paper a very fast feature selection technique based on conditional mutual information. By picking features which maximize their mutual information with the class to predict conditional to any feature already picked, it ensures the selection of features which are both individually informative and two-by-two weakly dependant. We show that this feature selection method outperforms other classical algorithms, and that a naive Bayesian classifier built with features selected that way achieves error rates similar to those of state-of-the-art methods such as boosting or SVMs. The implementation we propose selects 50 features among $40,000$, based on a training set of 500 examples in a tenth of a second on a standard 1Ghz PC.

Keywords: classification, mutual information, feature selection, naive Bayes, information theory, fast learning

1. Introduction

By reducing the number of features, one can both reduce overfitting of learning methods, and increase the computation speed of prediction (Guyon and Elisseeff, 2003). We focus in this paper on the selection of a few tens of binary features among a several tens of thousands in a context of classification.

Feature selection methods can be classified into two types, *filters* and *wrappers* (Kohavi and John, 1997; Das, 2001). The first kind are classifier agnostic, as they are not dedicated to a specific type of classification method. On the contrary the *wrappers* rely on the performance of one type of classifier to evaluate the quality of a set of features. Our main interest in this paper is to design an efficient *filter*, both from a statistical and from a computational point of view.

The most standard filters rank features according to their individual predictive power, which can be estimated by various means such as Fisher score (Furey et al., 2000), Kolmogorov-Smirnov test, Pearson correlation (Miyahara and Pazzani, 2000) or mutual information (Battiti, 1994; Bonnlander and Weigend, 1996; Torkkola, 2003). Selection based on such a ranking does not ensure weak dependency among features, and can lead to redundant and thus less informative selected families. To catch dependencies between features, a criterion based on decision trees has been proposed recently (Ratanamahatana and Gunopulos, 2003). Features which appear in binary trees build with the standard C4.5 algorithm are likely to be either individually informative (those at the top) or con-

ditionally informative (deeper in the trees). The drawbacks of such a method are its computational cost and sensitivity to overfitting.

Our approach iteratively picks features which maximize their mutual information with the class to predict, conditionally to the response of any feature already picked (Vidal-Naquet and Ullman, 2003; Fleuret, 2003). This Conditional Mutual Information Maximization criterion (CMIM) does not select a feature similar to already picked ones, even if it is individually powerful, as it does not carry additional information about the class to predict. Thus, this criterion ensures a good tradeoff between independence and discrimination. A very similar solution called Fast Correlation-Based Filter (Yu and Liu, 2003) selects features which are highly correlated with the class to predict if they are less correlated to any feature already selected. This criterion is very closed to ours but does not rely on a unique cost function which includes both aspects (i.e. information about the class and independence between features) and may be tricked in situation where the dependence between feature appears only conditionally on the object class. It also requires the tuning of a threshold δ for feature acceptance, while our algorithm does not.

Experiments demonstrate that CMIM outperforms the other feature selection methods we have implemented. Results also show and that a naive Bayesian classifier (Duda and Hart, 1973; Langley et al., 1992) based on features chosen with our criterion achieves error rates similar or lower than AdaBoost (Freund and Schapire, 1996a) or SVMs (Boser et al., 1992; Vapnik, 1998; Christiani and Shawe-Taylor, 2000). Also, experiments show the robustness of this method when challenged by noisy training sets. In such a context, it actually achieves better results than regularized AdaBoost, even though it does not require the tuning of any regularization parameter beside the number of features itself.

We also propose in this paper a fast but exact implementation based on a lazy evaluation of feature scores during the selection process. This implementation divides the computational time by two orders of magnitude and leads to a learning scheme which takes for instance one tenth of a second to select 50 out of $43,904$ features, based on 500 training examples. It is available under the GNU General Public Licence at `http://diwww.epfl.ch/~fleuret/files/cmim-1.0.tgz`.

In §2 we introduce the notation, summarize basic concepts of information theory, and present several feature selection schemes, including ours. In §3 we describe how features can be combined with standard classification techniques (perceptron, naive Bayesian, nearest neighbors and SVM with a Gaussian kernel). We propose a very fast implementation of our algorithm in §4 and give experimental results in §5 and §6. We finally analyze those results in §7 and §8.

2. Feature Selection

Our experiments are based on two tasks. The first one is a standard pattern recognition problem and consists of classifying small grey-scale pictures as *face* or *non face*, given a large number of elementary boolean edge-like features. The second one is a drug-design problem and consists of predicting the bio-activity of unknown molecules, given a large number of three-dimensional boolean properties. For clarity, we relate our notation to those two problems, but our approach is generic.

2.1 Notation

Denote by Y a boolean random variable standing for the real class of the object to classify. The value 1 (respectively 0) stands for *face* in the image classification experiments and for *active molecule* in

the drug design task (respectively for *non-face* and *inactive molecule*). Let $X_1, \ldots, X_N$ denote the N boolean features. They depend on the presence or absence of edges at certain positions in the image classification task and to certain three-dimensional properties of the molecule in the drug-design task (see §5.1.2 and §5.2).

The total number of features N is of the order of a few thousands. We denote by $X_{\nu(1)}, \ldots, X_{\nu(K)}$ the K features selected during the feature selection process. The number K of such features is very small compared to N, of the order of a few tens.

All statistical estimation is based on a few hundreds of samples, labeled by hand with their real classes and denoted

$$\mathfrak{L} \quad = \quad \{(x^{(1)}, y^{(1)}), \ldots, (x^{(T)}, y^{(T)})\}.$$

See §5.1 and §5.2 for a precise description of those sets. Each $x^{(t)} \in \{0,1\}^N$ is the boolean vector of feature responses on the tth training example. Hence, $x_n^{(t)}$ is the response of the nth feature on the sample t, and $x_n^{(1)}, \ldots, x_n^{(T)}$ are independent and identically distributed realizations of X_n. We denote $x_n \in \{0,1\}^T$ this vector of values of feature n on the training samples.

We will use those examples implicitly for all the empirical estimation during training.

2.2 Information Theory Tools

Information theory provides intuitive tools to quantify the uncertainty of random quantities, or how much information is shared by a few of them (Cover and Thomas, 1991). We consider in this section finite random variables and we denote by U, V and W The three of them.

The most fundamental concept in information theory is the entropy $H(U)$ of a random variable, which quantifies the uncertainty of U. The conditional entropy $H(U|V)$ quantifies the remaining uncertainty of U, when V is known. For instance, if U is a deterministic function of V, then this conditional entropy is zero, as no more information is required to describe U when V is known. On the contrary, if they are independent, knowing V does not tell you anything about U and the conditional entropy is equal to the entropy itself.

Our feature selection is based on the conditional mutual information

$$I(U;V|W) \quad = \quad H(U|W) - H(U|W,V).$$

This value is an estimate of the quantity of information shared between U and V when W is known. It can also be seen, as shown above, as the difference between the average remaining uncertainty of U when W is known and the same uncertainty when both W and V are known. If V and W carry the same information about U, the two terms on the right are equal, and the conditional mutual information is zero, even if both V and W are individually informative. On the contrary if V brings information about U which is not already contained in W the difference is large.

2.3 Conditional Mutual Information Maximization

The main goal of feature selection is to select a small subset of features that carries as much information as possible. The ultimate goal would be to choose $\nu(1), \ldots, \nu(K)$ which minimize $\hat{H}\left(Y|X_{\nu(1)}, \ldots, X_{\nu(K)}\right)$. But this expression can not be estimated with a training set of realistic size

as it requires the estimation of 2^{K+1} probabilities. Furthermore, even if there were ways to have an estimation, its minimization would be computationally intractable.

At the other extreme, one could do a trivial random sampling which would ensure to some extent independence between features (if different types of features are equally represented) but would not account for predictive power. This could be dealt with by basing the choice on an estimate of this predictive power. The main weakness of this approach is that although it takes care of individual performance, it does not avoid at all redundancy among the selected features. One would pick many similar features, as the ones carrying a lot of information are likely to be of a certain type. For face detection with edge-like features for instance, edges on the eyebrows and the mouth would be the only ones competitive as they are more face-specific than any other edge, yet numerous enough.

We propose an intermediate solution. Our approach deals with the tradeoff between individual power and independence by comparing each new feature with the ones already picked. We say that a feature X' is good only if $\hat{I}(Y\,;X'|X)$ is large for *every* X already picked. This means that X' is good only if it carries information about Y, and if this information has not been caught by any of the X already picked. More formally, we propose the following iterative scheme

$$\nu(1) = \arg\max_n \hat{I}(Y\,;X_n) \tag{1}$$

$$\forall k,\ 1\leq k<K,\quad \nu(k+1) = \arg\max_n \underbrace{\left\{\min_{l\leq k} \hat{I}\left(Y\,;X_n\,|\,X_{\nu(l)}\right)\right\}}_{s(n,k)}. \tag{2}$$

As said before, $\hat{I}\left(Y\,;X_n\,|\,X_{\nu(l)}\right)$ is low if either X_n does not bring information about Y or if this information was already caught by $X_{\nu(l)}$. Hence, the score $s(n,k)$ is low if at least one of the features already picked is similar to X_n (or if X_n is not informative at all).

By taking the feature X_n with the maximum score $s(n,k)$ we ensure that the new feature is both informative and different than the preceding ones, at least in term of predicting Y.

The computation of those scores can be done accurately as each score $I(Y\,;X_n\,|\,X_{\nu(l)})$ requires only estimating the distribution of triplets of boolean variables. Despite its apparent cost this algorithm can be implemented in a very efficient way. We will come back in details to such an implementation in §4.

Note that this criterion is equivalent to maximizing $I(X,X_{\nu(k)}\,;Y)-I(X_{\nu(k)}\,;Y)$, which is proposed in (Vidal-Naquet and Ullman, 2003).

2.4 Theoretical Motivation

In Koller and Sahami (1996) the authors propose using the concept of the Markov blanket to characterize features that can be removed without hurting the classification performance. A subfamily of features M is a blanket for a feature X_i if X_i is conditionally independent of the other feature and the class to predict given M. However, as the authors point out, such a criterion is stronger than what is really required which is the conditional independence between X_i and Y given M.

CMIM is a forward-selection of features based on an approximation of that criterion. This approximation considers families M composed of a unique feature already picked. Thus, a feature X can be discarded if there is one feature X_ν already picked such that X and Y are conditionally independent given X_ν. This can be expressed as $\exists k,\ I(Y\,;X\,|\,X_{\nu(k)})=0$. Since the mutual information is positive, this can be re-written

$$\min_k I(Y\,;X\,|\,X_{\nu(k)}) \quad = \quad 0.$$

Conversely, the higher this value, the more X is relevant. A natural criterion consists of ranking the remaining features according to that quantity, and to pick the one with the highest value.

2.5 Other Feature Selection Methods

This section lists the various feature selection methods we have used for comparison in our experiments.

2.5.1 RANDOM SAMPLING

The most trivial form of feature selection consist of a uniform random subsampling without repetition. Such an approach leads to features as independent as the original but does not pick the informative ones. This leads to poor results when only a small fraction of the features actually provide information about the class to predict.

2.5.2 MUTUAL INFORMATION MAXIMIZATION

To avoid the main weakness of the random sampling described above, we have also implemented a method which picks the K features $\nu(1), \ldots, \nu(K)$ maximizing individually the mutual information $\hat{I}\left(Y\,;X_{\nu(l)}\right)$ with the class to predict. Selection based on such a ranking does not ensure weak dependency among features, and can lead to redundant and poorly informative families of features.

In the following sections, we call this method MIM for Mutual Information Maximization.

2.5.3 C4.5 BINARY TREES

As proposed by Ratanamahatana and Gunopulos (2003), binary decision trees can be used for feature selection. The idea is to grow several binary trees and to rank features according to the number of times they appear in the top nodes. This technique is proposed in the literature as a good filter for naive Bayesian classifiers, and is a good example of a scheme able to spot statistical dependencies between more than two features, since the choice of a feature in a binary tree depends on the statistical behavior conditionally on the values of the ones picked above.

Efficiency was increased on our specific task by using randomization (Amit et al., 1997) which consist of using random subsets of the features instead of random subsets of training examples as in bagging (Breiman, 1999, 1996).

We have built 50 trees, each with one half of the features selected at random, and collected the features in the first five layers. Several configurations of number of trees, proportions of features and proportions of training examples were compared and the best one kept. This method is called "C4.5 feature selection" in the result sections.

2.5.4 FAST CORRELATION-BASED FILTER

Th FCBF method addresses explicitly the correlation between features. It first ranks the features according to their mutual information with the class to predict, and remove those which mutual information is lesser than a threshold δ.

In a second step, it iteratively removes any feature X_i if there exist a feature X_j such that $I(Y;X_j) \geq I(Y;X_i)$ and $I(X_i;X_j) \geq I(X_i;Y)$, i.e. X_j is better as a predictor of Y and X_i is more similar to X_j than to Y. The threshold δ can be adapted to get the expected number of features.

2.5.5 ADABOOST

A last method consists of keeping the features selected during boosting and is described precisely in §3.4, page 1538.

3. Classifiers

To evaluate the efficiency of the CMIM feature selection method, we compare the error rates of classifiers based on the features it selects to the error rates with the same classifiers build on features selected by other techniques.

We have implemented several classical type of classifiers, two linear (perceptron and naive Bayesian) and two non-linear (k-NN and SVM with a Gaussian kernel), to test the generality of CMIM. We also implemented a boosting method which avoids the feature selection process since it selects the features and combine them into a linear classifier simultaneously. This technique provides a baseline for classification score.

In this section, recall that we denote by $X_{\nu(1)}, \ldots, X_{\nu(K)}$ the selected features.

3.1 Linear Classifiers

A linear classifier depends on the sign of a function of the form

$$f(x_1, \ldots, x_N) \quad = \quad \sum_{k=1}^{K} \omega_k x_{\nu(k)} \; + \; b.$$

We have used two algorithms to estimate the $(\omega_1, \ldots, \omega_K)$ and b from the training set $\mathfrak{L}$. The first one is the classical perceptron (Rosenblatt, 1958; Novikoff, 1962) and the second one is the naive Bayesian classifier (Duda and Hart, 1973; Langley et al., 1992).

3.1.1 PERCEPTRON

The perceptron learning scheme estimates iteratively the normal vector $(\omega_1, \ldots, \omega_K)$ by correcting it as long as training examples are misclassified. More precisely, as long as there exists a misclassified example, its feature vector is added to the normal vector if it is of class positive, and is otherwise subtracted. The bias term b is computed by considering a constant feature always "positive". If the training set is linearly separable in the feature space, the process is known to converge to a separating hyperplane and the number of iterations can be easily bounded (Christiani and Shawe-Taylor, 2000, page. 12–14). If the data are not linearly separable, the process is terminated after an a priori fixed number of iterations.

Compared to linear SVM for instance, the perceptron has a greater algorithmic simplicity, and suffers from different weaknesses (over-fitting in particular). It is an interesting candidate to estimate the quality of the feature selection methods as a way to control overfitting.

3.1.2 NAIVE BAYESIAN

The naive Bayesian classifier is a simple likelihood ratio test with an assumption of conditional independence among the features. The predicted class depends on the sign of

$$f(x_1,\ldots,x_N) \quad = \quad \log \frac{\hat{P}(Y=1\,|\,X_{\nu(1)}=x_{\nu(1)},\ldots,X_{\nu(K)}=x_{\nu(K)})}{\hat{P}(Y=0\,|\,X_{\nu(1)}=x_{\nu(1)},\ldots,X_{\nu(K)}=x_{\nu(K)})}.$$

Under the assumption that the $X_{\nu(.)}$ are conditionally independent, given Y, and with $a=\log\frac{\hat{P}(Y=1)}{\hat{P}(Y=0)}$, we have

$$\begin{aligned} f(x_1,\ldots,x_N) &= \log \frac{\prod_{k=1}^{K}\hat{P}(X_{\nu(k)}=x_{\nu(k)}\,|\,Y=1)}{\prod_{k=1}^{K}\hat{P}(X_{\nu(k)}=x_{\nu(k)}\,|\,Y=0)} \;+\; a \\ &= \sum_{k=1}^{K}\log\frac{\hat{P}(X_{\nu(k)}=x_{\nu(k)}\,|\,Y=1)}{\hat{P}(X_{\nu(k)}=x_{\nu(k)}\,|\,Y=0)} \;+\; a \\ &= \sum_{k=1}^{K}\left\{\log\frac{\hat{P}(X_{\nu(k)}=1\,|\,Y=1)}{\hat{P}(X_{\nu(k)}=1\,|\,Y=0)}\frac{\hat{P}(X_{\nu(k)}=0\,|\,Y=0)}{\hat{P}(X_{\nu(k)}=0\,|\,Y=1)}\right\}x_{\nu(k)} \;+\; b. \end{aligned}$$

Thus we finally obtain a simple expression for the coefficients

$$\omega_k \quad = \quad \log\frac{\hat{P}(X_{\nu(k)}=1\,|\,Y=1)}{\hat{P}(X_{\nu(k)}=1\,|\,Y=0)}\frac{\hat{P}(X_{\nu(k)}=0\,|\,Y=0)}{\hat{P}(X_{\nu(k)}=0\,|\,Y=1)}.$$

The bias b can be estimated empirically given the ω_k to minimize the error rate on the training set.

3.2 Nearest Neighbors

We used the Nearest-Neighbors as a first non-linear classifier. Given a regularization parameter k and an example x, the k-NN technique considers the k training examples closest to x according to their distance in the feature space $\{0,1\}^K$ (either L^1 or L^2, which are equivalent in that case), and gives as predicted class the dominant real class among those k examples. To deal with highly unbalanced population, such as in drug-design series of experiments, we have introduced a second parameter $\alpha\in[0,1]$ used to weight the positive examples.

Both k and α are optimized by cross-validation during training.

3.3 SVM

As a second non-linear technique, we have used a SVM (Boser et al., 1992; Vapnik, 1998; Christiani and Shawe-Taylor, 2000) based on a Gaussian kernel. This technique is known to be robust to overfitting and has demonstrated excellent behavior on a very large spectrum of problems. The unbalanced population is dealt with by changing the soft-margin parameters C accordingly to the number of training samples of each class, and the choice of the σ parameter of the kernel is described in §6.1.

3.4 AdaBoost

The idea of boosting is to select and combine several classifiers (often referred to as *weak learners*, as they may have individually high error rate) into an accurate one with a voting procedure. In our case, the finite set of features is considered as the space of weak learners itself (i.e. each feature is considered as a boolean predictor). Thus, the training of a weak learner simply consist of picking the one with the minimum error rate. We allow negative weights, which is equivalent to adding for any feature X_n its anti-feature $1 - X_n$. Note that this classifier is not combined with a feature selection methods since it does both the feature selection and the estimation of the weights to combine them simultaneously.

The process maintains a distribution on the training examples which concentrates on the misclassified ones during training. At iteration k, the feature $X_{\nu(k)}$ which minimizes the weighted error rate is selected, and the distribution is refreshed to increase the weight of the misclassified samples and reduce the importance of the others. Boosting can be seen as a functional gradient descent (Breiman, 2000; Mason et al., 2000; Friedman et al., 2000) in which each added weak learner is a step in the space of classifiers. From that perspective, the weight of a given sample is proportional to the derivative of the functional to minimize with respect to the response of the result classifier on that sample: the more a correct prediction on that particular example helps to optimize the functional, the higher its weight.

In our comparisons, we have used the original AdaBoost procedure (Freund and Schapire, 1996a,b), which is known to suffer from overfitting. For noisy tasks, we have chosen a soft-margin version called AdaBoost$_{reg}$ (Ratsch et al., 1998), which regularizes the classical AdaBoost by penalizing samples which too heavily influence the training, as they are usually outliers.

To use boosting as a feature selector, we just keep the set of selected features $X_{\nu(1)}, \ldots, X_{\nu(K)}$, and combine them with another classification rule instead of aggregating them linearly with the weights $\omega_1, \ldots, \omega_K$ computed during the boosting process.

4. CMIM Implementations

We describe in this section how we compute efficiently entropy and mutual information and give both a naive and an efficient implementation of CMIM.

4.1 Mutual Information Estimation

For the clarity of the algorithmic description that follows, we describe here how mutual information and conditional mutual information are estimated from the training set. Recall that as said in §2.1 for any $1 \leq n \leq N$ we denote by $x_n \in \{0,1\}^T$ the vector of responses of the nth feature on the T training samples.

The estimation of the conditional entropy, mutual information, or conditional mutual information can be done by summing and subtracting estimation of entropies of families of one to three variables. Let x, y, and z be three boolean vectors and u, v, and w, three boolean values. We denote by $||.||$ the cardinal of a set and define three counting functions

$$\begin{aligned} \eta_u(x) &= ||\{t : x^{(t)} = u\}|| \\ \eta_{u,v}(x,y) &= ||\{t : x^{(t)} = u,\, y^{(t)} = v\}|| \end{aligned}$$

$$\eta_{u,v,w}(x,y,z) \quad = \quad ||\{t : x^{(t)} = u, y^{(t)} = v, z^{(t)} = w\}||.$$

From this, if we define $\forall x, \xi(x) = \frac{x}{T}\log(x)$, with the usual convention $\xi(0) = 0$, we have

$$\begin{aligned}
\hat{H}(Y) &= \log(T) - \sum_{u \in \{0,1\}} \xi(\eta_u(y)) \\
\hat{H}(Y, X_n) &= \log(T) - \sum_{u,v \in \{0,1\}^2} \xi(\eta_{u,v}(y, x_n)) \\
\hat{H}(Y, X_n, X_m) &= \log(T) - \sum_{u,v,w \in \{0,1\}^3} \xi(\eta_{u,v,w}(y, x_n, x_m)).
\end{aligned}$$

And by definition, we have

$$\begin{aligned}
\hat{I}(Y\,;X_n) &= \hat{H}(Y) + \hat{H}(X_n) - \hat{H}(Y, X_n) \\
\hat{I}(Y\,;X_n\,|\,X_m) &= \hat{H}(Y\,|\,X_m) - \hat{H}(Y\,|\,X_n, X_m) \\
&= \hat{H}(Y, X_m) - \hat{H}(X_m) - \hat{H}(Y, X_n, X_m) + \hat{H}(X_n, X_m).
\end{aligned}$$

Finally, those computations are based on counting the numbers of occurrences of certain patterns of bits in families of one to three vectors, and evaluations of ξ on integer values between 0 and T. The most expensive operation is the former: the evaluations of the η_u, $\eta_{u,v}$ and $\eta_{u,v,w}$, which can be decomposed into bit counting of conjunctions of binary vectors. The implementation can be optimized by using a look-up table to count the number of bits in couples of bytes and computing the conjunctions by block of 32 bits. Also, note that because the evaluation of ξ is restricted on integers smaller than T, a lookup table can be used for it too. In the pseudo-code, the function `mut_inf`(n) computes $\hat{I}(Y\,;X_n)$ and `cond_mut_inf`(n,m) computes $\hat{I}(Y\,;X_n\,|\,X_m)$.

Note that the naive Bayesian coefficients can be computed very efficiently with the same counting procedures

$$\omega_k \quad = \quad \log \eta_{0,0}(x_{\nu(k)}, y) + \log \eta_{1,1}(x_{\nu(k)}, y) - \log \eta_{1,0}(x_{\nu(k)}, y) - \log \eta_{0,1}(x_{\nu(k)}, y).$$

4.2 Standard Implementation

The most straight-forward implementation of CMIM keeps a score vector s which contains for every feature X_n, after the choice of $\nu(k)$, the score $s[n] = \min_{l \leq k} \hat{I}\left(Y\,;X_n \mid X_{\nu(l)}\right)$. This score table is initialized with the values $\hat{I}(Y\,;X_n)$.

The algorithm picks at each iteration the feature $\nu(k)$ with the highest score, and then refreshes every score $s[n]$ by taking the minimum of $s[n]$ and $\hat{I}\left(Y\,;X_n \mid X_{\nu(k)}\right)$. This implementation is given in pseudo-code on Algorithm 1 and has a cost of $O(K \times N \times T)$.

4.3 Fast Implementation

The most expensive part in the algorithm described above are the $K \times N$ calls to `cond_mut_inf`, each costing $O(T)$ operations. The fast implementation of CMIM relies on the fact that because the

Algorithm 1 Simple version of CMIM

for $n = 1 \dots N$ **do**
  $s[n] \leftarrow \texttt{mut\_inf}(n)$
for $k = 1 \dots K$ **do**
  $nu[k] = \arg\max_n s[n]$
  for $n = 1 \dots N$ **do**
    $s[n] \leftarrow \min(s[n], \texttt{cond\_mut\_inf}(n, nu[k]))$

score vector can only decrease when the process goes on, bad scores may not need to be refreshed. This implementation does not rely on any approximation and produces the exact same results as the naive implementation described above.

Intuitively, consider a set of features containing several ones almost identical. Picking one of them makes all the other ones of this group useless during the rest of the computation. This can be spotted early because their scores are low, and will remain so because scores can only decrease.

The fast version of CMIM stores for every feature X_n a partial score $ps[n]$, which is the minimum over a few of the conditional mutual informations appearing in the min in equation (2) page 1534. Another vector $m[n]$ contains the index of the last picked feature taken into account in the computation of $ps[n]$. Thus, we have at any moment

$$ps[n] = \min_{l \leq m[n]} \hat{I}\left(Y \,;\, X_n \mid X_{v(l)}\right).$$

At every iteration, the algorithm goes through all candidates and update its score only if the best one found so far in that iteration is not better, since scores can only go down when updated. For instance, if the *up-to-date score* of the first feature was 0.02 and the *non-already updated* score of the second feature was 0.005, it is not necessary to update the later, since it can only go down.

The pseudo-code on Algorithm 2 is an implementation of that idea. It goes through all the candidate features, but does not compute the conditional mutual information between a candidate and the class to predict, given the most recently picked features, if the score of that candidate is below the best up-to-date score $s^\star$ found so far in that iteration (see figure 1).

Algorithm 2 Fast version of CMIM

for $n = 1 \dots N$ **do**
  $ps[n] \leftarrow \texttt{mut\_inf}(n)$
  $m[n] \leftarrow 0$
for $k = 1 \dots K$ **do**
  $s^\star \leftarrow 0$
  for $n = 1 \dots N$ **do**
    while $ps[n] > s^\star$ **and** $m[n] < k-1$ **do**
      $m[n] \leftarrow m[n] + 1$
      $ps[n] \leftarrow \min(ps[n], \texttt{cond\_mut\_inf}(n, nu[m[n]]))$
    if $ps[n] > s^\star$ **then**
      $s^\star \leftarrow ps[n]$
      $nu[k] \leftarrow n$

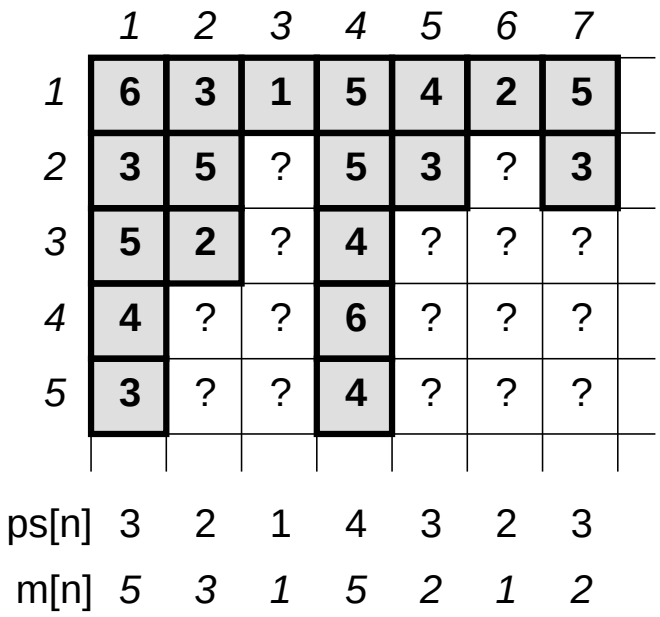


Figure 1: *The cell in column n and row l in the array contains the value* `cond_mut_inf`$(n, nu[l])$. *The score of feature X_n at step $k+1$ is the minimum over the k top-cells of column n. While the naive version evaluates the values of all cells in the first k rows, the fast version computes a partial score, which is the minimum over only the first $m[n]$ cells in column n. It does not update a feature score $ps[n]$ if its current value is already below the best score of a column found so far in that iteration.*

5. Experimental Settings

All the experiments have been done with softwares written in C++ on GNU/Linux computers. We have used free software tools (editor, compiler, debugger, word-processors, etc.), mainly from the Free Software Foundation.[1] We have also used the Libsvm[2] for the SVM.

5.1 Image Classification

This task is a classical pattern-recognition problem in which one tries to predict the real class of a picture. The input data are small grayscale patches and the two classes are face and background (non-face) pictures.

5.1.1 TRAINING AND TEST SETS

We have used training and test sets built from two large sets of scenes. Those original big sets were assembled by collecting a few thousand scenes from the web and marking by hand the locations of eyes and mouth on every visible frontal viewed face. Using two sets ensures that examples belonging to the same scene series will all be used either as training pictures or as test pictures. This prevents from trivial similarities between the training and test examples.

From every face of every scene we generate ten small grayscale face images by applying rotation, scaling and translation to randomize its pose in the image plan. We have also collected complex scenes (forests, buildings, furniture, etc.) from which we have automatically extracted tens of thousands of background (non-face) pictures. This leads to a total of $14,268$ faces and $14,800$ background pictures for training (respectively $5,202$ and $5,584$ for test).

All those images, faces and backgrounds, are of size 28×28 pixels, and with 256 grayscale levels. Faces have been registered roughly so that the center of the eyes is in a 2×2 central square, the distance between the eyes is between 10 to 12 pixels and the tilt is between -20 and $+20$ degrees (see figure 2 for a few examples of images).

For each experiment both the training and the test sets contain 500 images, roughly divided into faces and non-faces. Errors are averaged over 25 rounds with such training and test sets.

5.1.2 EDGE FEATURES

We use features similar to the edge fragment detectors in (Fleuret and Geman, 2001, 2002). They are easy to compute, robust to illumination variations, and do not require any tuning.

Each feature is a boolean function indexed by a location (x,y) in the 28×28 reference frame of the image, a direction d which can take 8 different values (see figures 3 and 4) and a tolerance t which is an integer value between 1 and 7 (this maximum value has been fixed empirically). The tolerance corresponds to the size of the neighborhood where the edge can "float", i.e. where it has to be present for the feature to be equal to 1 (see figure 3).

For every location in the reference frame (i.e. every pixel), we thus have 7×8 features, one for each couple direction / tolerance. For tolerance 1, those features are simple edge fragment detector (see figure 4). For tolerance 2, they are disjunction of those edge fragment detectors on two locations for each pixel, etc.

The total number of such features is $28 \times 28 \times 8 \times 7 = 43,904$.

1. http://www.fsf.org
2. http://www.csie.ntu.edu.tw/~cjlin/libsvm

Figure 2: The two upper rows show examples of background pictures, and the two lower rows show examples of face pictures. All images are grayscale of size 28×28 pixels, extracted from complete scenes found on the World Wide Web. Faces are roughly centered and standardized in size.

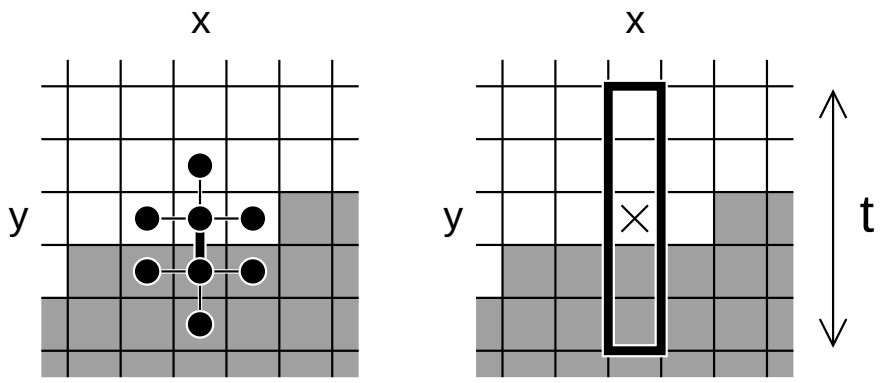


Figure 3: The boolean features we are using are crude edge detectors, invariant to changes in illumination and to small deformations of the image. The picture on the left shows the criterion for a horizontal edge located in (x,y). The detector responds positively if the six differences between pixels connected by a thin segment are lesser in absolute value than the difference between the pixels connected by the thick segment. The relative values of the two pixels connected by the thick line define the polarity of the edge (dark to light or light to dark). The picture on the right shows the strip where the edge can "float" for the feature to respond when the tolerance t is equal to 5.

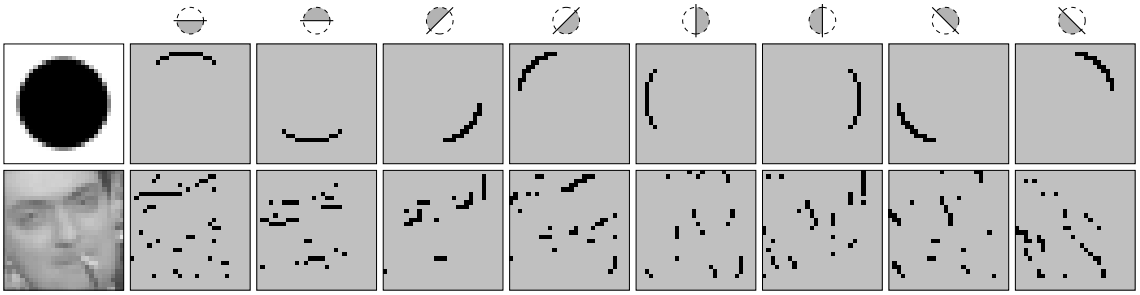


Figure 4: The original grayscale pictures are shown on the left. The eight binary maps on the right show the responses of the edge detectors at every locations in the 28×28 frame, for every one of the 8 possible directions and polarities. The binary features are disjunctions (ORings) of such edge detectors in small neighborhoods, which ensure their robustness to image deformations.

5.2 Prediction of Molecular Bio-activity

The second data set is based on $1,909$ compounds tested for their ability to bind to a target site on thrombin. This corresponds to a drug-design task in which one tries to predict which molecules will achieve the expected effect.

Each compound has a binary class (*active* or *inactive*) and $139,351$ binary features standing for as many three-dimensional properties. The exact semantic of those features remains unknown but is consistent among the samples. To be able to use many techniques in our comparisons, we restricted the number of binary features to $2,500$ by a rough random sampling, since the computation time would have been intractable with classical methods on such a large number of binary features.

All the experiments are done with 25 rounds of cross-validation. For each one of this round, 100 samples are randomly picked as test examples, and all the others used for training. Since the population are highly unbalanced (42 positive vs. 1867 negative examples), the balanced error rate (average of false positive and false negative error rates) was used both for training and testing.

The dataset was provided by DuPont Pharmaceutical for the KDD-Cup 2001 competition[3] and was used in 2003 for the NIPS feature selection challenge[4] under the name DOROTHEA.

6. Results

The experimental results we present in this section address both performance in term of error rates and speed.

In §6.1, we compare several associations of a feature selection method (CMIM, MIM, C4.5, random and AdaBoost as a feature selection method) and a classifier (naive Bayesian, *k*-NN, Gaussian SVM, perceptron). Also, AdaBoost and a regularized version of AdaBoost were tested on the same data.

For the image recognition task, since the error rates with 50 features correspond roughly to the asymptotic score of our main methods (CMIM + naive Bayesian and AdaBoost, see figure 5), we

3. `http://www.cs.wisc.edu/~dpage/kddcup2001/`
4. `http://www.nipsfsc.ecs.soton.ac.uk`

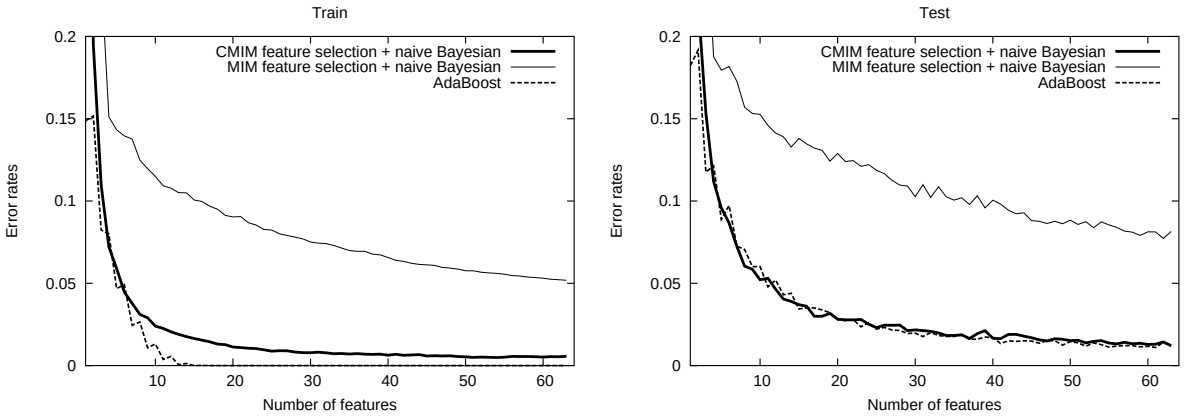


Figure 5: The asymptotic error rates are reached with 50 features on the picture classification task.

have used this number of features for the extensive comparisons. Similarly, the number of features was 10 for the bio-activity prediction.

The σ parameter of the Gaussian kernel was chosen separately for every feature selection method by optimizing the test error with a first series of 25 rounds. The training and test errors reported in the results section are estimated by running 25 other rounds of cross-validation. The results may suffer slightly from over-fitting and over-estimate the score of the SVM. However the effect is likely to be negligible considering the large size of the complete sets.

In §6.2 we compare the fast implementation of CMIM to the naive one and provide experimental computation times in the task of image recognition.

6.1 Error Rates

To quantify the statistical significance in our comparisons, we estimate empirical error rates form the data sets, but also the empirical variance of those estimates. Those variances are computed under the assumption that the samples are independent and identically distributed.

We provide for every experiments in the tables 1, 2 and 3 both the estimated test error e and the score $\frac{e^\star - e}{\sqrt{\sigma_{e^\star} + \sigma_e}}$ where $e^\star$ is the score of our reference setting (CMIM + Naive Bayesian), and $\sigma_{e^\star}$ and σ_e are the empirical variances of the error rate estimates. This empirical variance are estimated simply as empirical variance of Bernoulli variables.

6.1.1 IMAGE CLASSIFICATION

The first round of experiments uses the dataset of pictures described in §5.1. The results on table 1 show that the best scores are obtained with CMIM + SVM, closely followed by AdaBoost features combined also with SVM. The Naive Bayesian with CMIM features performs pretty well, ranking fourth. CMIM as a feature selection method is always the best, for any given classification technique.

It is meaningful to note that the computational cost of SVM is few orders of magnitudes higher than those of AdaBoost alone or CMIM + Bayesian as it requires for training the computation of the optimal σ through cross-validation, and requires during classification the evaluation of hundreds of exponentials.

Classifier	Training error	Test error (e)	$\frac{e^\star - e}{\sqrt{\sigma_{e^\star} + \sigma_e}}$
CMIM + SVM	0.53%	1.12%	-2.77
AdaBoost feature selection + SVM	0%	1.21%	-2.11
AdaBoost	0%	1.45%	-0.45
CMIM feature selection + naive Bayesian	0.52%	1.52%	–
CMIM feature selection + k-NN	0%	1.69%	1.07
AdaBoost feature selection + k-NN	0%	1.71%	1.19
FCBF feature selection + SVM	0.75%	1.85%	2.02
FCBF feature selection + naive Bayesian	1.28%	2.13%	3.60
CMIM feature selection + perceptron	0%	2.28%	4.40
AdaBoost feature selection + perceptron	0%	2.46%	5.32
C4.5 feature selection + SVM	0.73%	2.58%	5.91
FCBF feature selection + k-NN	0%	2.75%	6.73
C4.5 feature selection + perceptron	0%	3.26%	9.02
C4.5 feature selection + naive Bayesian	1.4%	3.28%	9.11
FCBF feature selection + perceptron	0%	3.50%	10.03
AdaBoost feature selection + naive Bayesian	0.4%	3.51%	10.06
C4.5 feature selection + k-NN	0%	3.57%	10.31
MIM + SVM	3.26%	5.67%	17.73
MIM feature selection + perceptron	3.56%	8.28%	25.06
MIM feature selection + naive Bayesian	5.58%	8.54%	25.72
MIM feature selection + k-NN	0.23%	8.99%	26.84
Random feature selection + SVM	9.04%	11.86%	33.44
Random feature selection + perceptron	13.36%	17.45%	44.66
Random feature selection + k-NN	0.30%	21.54%	52.18
Random feature selection + naive Bayesian	21.69%	24.77%	57.93

Table 1: Error rates with 50 features on the accurate face vs. background data set. The right column shows the difference between the test error rate $e^\star$ of the CMIM + naive Bayesian method and the test error rate e in the given row, divided by the standard deviation of that difference.

To test the robustness of the combination of CMIM and naive Bayes, we have run a second round of experiments with noisy training data, known to be problematic for boosting schemes. We generated the new training set by flipping at random 5% of the training labels. This corresponds to a realistic situation in which some training examples have been mis-labelled.

It creates a difficult situation for learning methods which take care of outliers, since there are 5% of them, distributed uniformly among the training population. Results are summarized in table 2. Note that the performance of the regularized version of AdaBoost correspond to the optimal performance on the *test set*.

All methods based on perceptron or boosting have high error rates, since they are very sensitive to outliers. The best classification techniques are those protected from over-fitting, thus SVM, Naive Bayesian and regularized AdaBoost, which take the 8 first rankings. Again in this experiment, CMIM is the best feature selection method for any classification scheme.

The FCBF method, which is related to CMIM since it looks for features both highly correlated with the class to predict and two-by-two uncorrelated scores very well, better than in the non-noisy case. It may be due to the fact that in this noisy situation protection from overfitting matters more than picking optimal features on the training set.

6.1.2 MOLECULAR BIO-ACTIVITY

This third round of experiments is more difficult to analyze since the characteristics of the features we deal with are mainly unknown. Because of the highly unbalanced population, methods sensitive to overfitting perform badly.

Results for these experiments are given 3. Except in one case (SVM), CMIM leads to the lowest error rate for any classification method. Also, when combined with the naive Bayesian rule, it gets lower error rates than SVM or nearest-neighbors.

The same dataset was used in the NIPS 2003 challenge,[5] in which it was divided in three subsets for training, test and validation (respectively of size 800, 350 and 800). Our main method CMIM + Bayesian achieves 12.46% error rate on the validation set without any tunning, while the top-ranking method achieves 5.47% with a Bayesian Network, see (Guyon et al., 2004) for more details on the participants and results.

6.2 Speed

The image classification task requires the selection of 50 features among $43,904$ with a training set of 500 examples. The naive implementation of CMIM takes 18800ms to achieve this selection on a standard 1Ghz personal computer, while with the fast version this duration drops to 255ms for the exact same selection, thus a factor of 73. For the thrombin dataset (selecting 10 features out of $139,351$ based on $1,909$ examples) the computation times drops from $156,545$ms with the naive implementation to $1,401$ms with the fast one, corresponding to a factor 110.

The dramatic gain in performance can be explained by looking at the number of calls to `cond_mut_inf`, which drops for the faces by a factor 80 (from $4,346,496$ to $54,928$), and for the thrombin dataset by a factor 220 (from $13,795,749$ to $62,125$).

The proportion of calls to `cond_mut_inf` for the face dataset is depicted on figure 6. We have also looked at the number of calls required to sort out each feature. In the simple implementation that number is the same for all features and is equal to the number of selected features K. For the fast

5. `http://www.nipsfsc.ecs.soton.ac.uk`

Classifier	Training error	Test error (e)	$\frac{e^\star - e}{\sqrt{\sigma_{e^\star} + \sigma_e}}$
CMIM + SVM	5.68%	1.37%	-3.59
FCBF feature selection + SVM	6.02%	1.49%	-2.79
CMIM feature selection + naive Bayesian	5.06%	1.95%	–
FCBF feature selection + naive Bayesian	5.38%	2.39%	2.38
C4.5 + SVM	5.57%	2.99%	5.30
AdaBoost$_{reg}$ (optimized on test set)	3.80%	3.06%	5.61
C4.5 feature selection + naive Bayesian	6.14%	3.62%	8.03
AdaBoost feature selection + SVM	4.39%	4.18%	10.25
CMIM feature selection + k-NN	0.08%	5.36%	14.42
MIM + SVM	7.85%	5.87%	16.07
AdaBoost	0.58%	6.33%	17.48
C4.5 feature selection + k-NN	0.71%	6.34%	17.52
FCBF feature selection + k-NN	0.87%	6.50%	17.99
AdaBoost feature selection + k-NN	0.39%	7.20%	20.02
AdaBoost feature selection + perceptron	0.12%	8.23%	22.82
MIM feature selection + naive Bayesian	9.47%	8.59%	23.75
CMIM feature selection + perceptron	7.36%	9.32%	25.60
FCBF feature selection + naive Bayesian	8.20%	9.33%	25.62
AdaBoost feature selection + naive Bayesian	10.28%	9.46%	25.94
C4.5 feature selection + perceptron	7.58%	11.06%	29.71
Random + SVM	13.00%	12.19%	32.23
MIM feature selection + k-NN	2.92%	11.46%	30.61
MIM feature selection + perceptron	11.53%	13.12%	34.23
Random feature selection + perceptron	19.47%	20.58%	48.75
Random feature selection + k-NN	1.43%	24.77%	56.29
Random feature selection + naive Bayesian	24.13%	24.99%	56.68

Table 2: Error rates with 50 features on the face vs. background data set whose training labels have been flipped with probability 5%. The right column shows the difference between the test error rate $e^\star$ of the CMIM + naive Bayesian method and the test error rate e in the given row, divided by the standard deviation of that difference.

Classifier	Training error	Test error (e)	$\frac{e^\star - e}{\sqrt{\sigma_{e^\star} + \sigma_e}}$
CMIM feature selection + naive Bayesian	10.45%	11.72%	–
AdaBoost feature selection + SVM	9.35%	12.99%	1.36
AdaBoost feature selection + naive Bayesian	10.29%	13.60%	1.99
AdaBoost$_{reg}$ (optimized on test set)	9.48%	13.64%	2.04
CMIM + SVM	13.21%	13.65%	2.05
AdaBoost	9.49%	13.76%	2.16
C4.5 feature selection + naive Bayesian	9.22%	13.90%	2.31
C4.5 + SVM	8.72%	17.34%	5.65
CMIM feature selection + k-NN	17.17%	18.77%	6.97
FCBF feature selection + naive Bayesian	13.62%	19.22%	7.37
FCBF feature selection + SVM	13.39%	23.14%	10.76
MIM feature selection + naive Bayesian	21.53%	23.35%	10.94
CMIM feature selection + perceptron	20.31%	23.51%	11.08
C4.5 feature selection + perceptron	12.86%	23.88%	11.38
MIM + SVM	24.65%	25.75%	12.93
FCBF feature selection + perceptron	21.98%	27.06%	13.98
FCBF feature selection + k-NN	19.28%	27.94%	14.68
Random + SVM	30.10%	30.92%	17.05
C4.5 feature selection + k-NN	24.18%	34.11%	19.54
Random feature selection + naive Bayesian	39.32%	40.13%	24.23
MIM feature selection + perceptron	32.70%	40.27%	24.34
Random feature selection + perceptron	43.61%	45.68%	28.63
Random feature selection + k-NN	45.09%	47.29%	29.94
MIM feature selection + k-NN	50.00%	50.00%	32.19

Table 3: Error rates with 10 features on the Thrombin dataset. The right column shows the difference between the test error rate $e^\star$ of the CMIM + naive Bayesian method and the test error rate e in the given row, divided by the standard deviation of that difference.

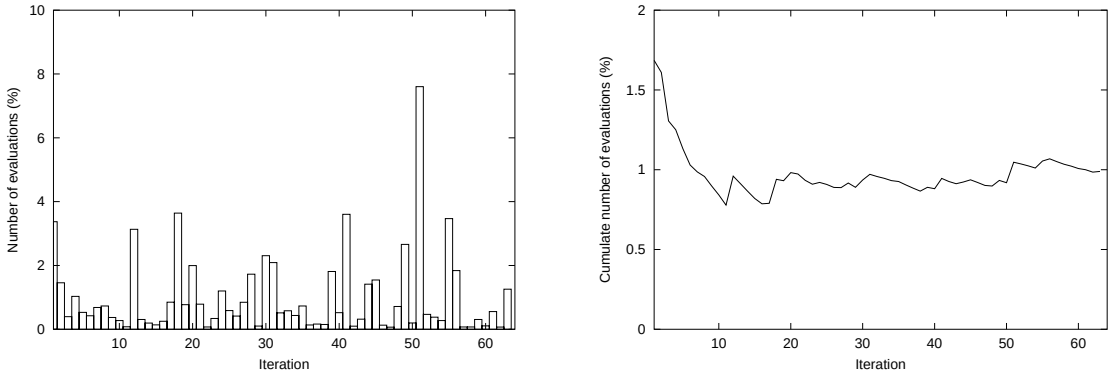


Figure 6: *Those curves show the proportion of calls to* `cond_mut_inf` *actually done in the fast version compared to the standard version for each iteration. The curve on the left shows the proportion for each step of the selection process, while the curve on the right shows the proportion of cumulate evaluations since the beginning. As it can be seen, this proportion is around* 1% *on the average.*

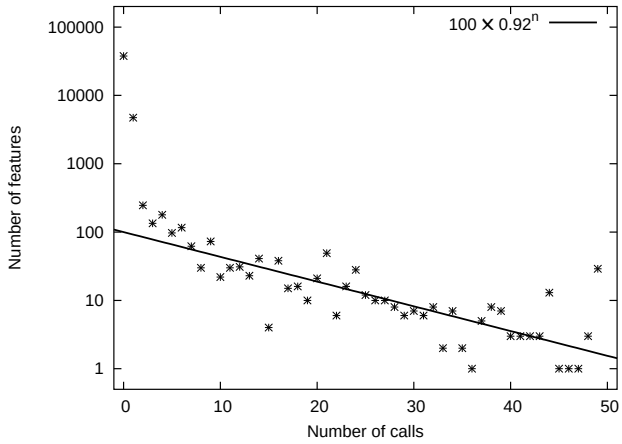


Figure 7: *This curve shows on an logarithmic scale how many features (y axis) require a certain number of calls to* `cond_mut_inf` *(x axis). The peak on the right corresponds to the* 50 *features actually selected, which had to be compared with all the other features, thus requiring* 49 *comparisons.*

version, this number depends on the feature, as very inefficient ones will probably require only one of them. The distribution of the number of evaluations is represented on figure 7 on a logarithmic scale, and fits roughly 100×0.92^n. This means that there are roughly 8% fewer features which require $n+1$ evaluations than features which require n evaluations.

7. Discussion

The experimental results we provide show the strength of the CMIM, even when combined with a simple naive Bayesian rule, since it ranks 4th, 3rd and 1st in the three experiments in which it is compared with 26 other combination feature selection + classifier.

Classification Power

It is easy to build a task CMIM can not deal with. Consider a situation where the positive population is a mixture of two sub-populations, and where half of the features provide information about the first population, while the other features provide information about the second population. This can happen in an image context by considering two different objects which do not share informative edges.

In such a situation, if one sub-population dominates statistically, CMIM does not pick feature providing information about the second sub-population. It would go on picking feature informative about the domineering sub-population as long as independent features remain.

A feature selection based on C4.5 would be able to catch informative features since the minority class would quickly be revealed as the source of uncertainty, and features dedicated to them would be selected. Similarly, AdaBoost can handle such a challenge because the error concentrates quickly on the second sub-population, which eventually drives the choice of features. In fact, both can be ween as wrappers since they take into account the classification outcome to select the features.

We could fix this weakness of CMIM by weighting challenging examples as well, forcing the algorithm to care about problematic minorities and pick features related to them. This would be the dual solution to AdaBoost regularization techniques which on the contrary reduce the influence of outliers.

From that point of view, CMIM and AdaBoost are examples of two families of learning methods. The first one is able to cope with overfitting by making a strong assumption of homogeneity of the informative power of features, while the second one is able to deal with a composed population by sequentially focusing on sub populations as soon as they become the source of error.

In both cases, the optimal tradeoff has to be specified on a per-problem basis, as there is no absolute way to know if the training examples are reliable examples of a complex mixture or noisy examples of an homogeneous population.

Speed

CMIM and boosting share many similarities from an algorithmic point of view. Both of them pick features one after another and compute at each iteration a score for every single candidate which requires to go through every training example.

Despite this similarity the lazy evaluation idea can not be applied directly to boosting. One could try to estimate a bound of the score of a weak learner at iteration $k+1$ (which is a weighted error rate), given its value at iteration k, using a bound on the weight variation. Practically, this idea gives very bad results because the variation can not be controlled efficiently and turns out to be very pessimistic. It leads to a negligible rate of rejection of the candidate features to check.

The feature selection based on C4.5 is even more difficult to optimize. Because of the complex interactions between features selected in previous nodes and the remaining candidates at a given

node, there is no simple way to predict that a feature can be ignored without reducing the performance of the method.

Usability

The CMIM algorithm does not require the tuning of any regularizing parameter, and since the implementation is an exact exhaustive search it also avoids the tuning of an optimization scheme.

Also, compared to methods like SVM or AdaBoost, both the feature selection criterion and the naive Bayesian classifier have a very clear statistical semantic. Since the naive Bayesian is an approximation of a likelihood ratio test, it can be easily combined with other techniques such as HMM, Bayesian Inference and more generally with other statistical methods.

Multi-class, Continuous Valued Features and Regression

Extension to the multi-class problem can be addressed either with a classifier-agnostic technique (for instance training several classifiers dedicated to different binary problems which can be combined into a multi-class predictor (Hastie and Tibshirani, 1998) or by extending CMIM and the Bayesian classifier directly to the multi-class context.

In that case, the price to pay is both in term of accuracy and computation cost. The estimation of the conditional mutual information requires the estimation of the empirical distribution of triples of variables, or N_c^3 empirical probabilities in a N_c class problem. Thus, accurate estimation requires as many more training samples. From the implementation perspective the fast version can be kept as-is but the computation of a conditional mutual information is $O(N_c^3)$, and the boolean computations by block require a $O(N_c)$ memory usage.

Extension to the case of continuous valued features and to regression (continuous valued class) is the center of interest of our current works. It is natural as soon as parametric density models are provided for any variable, couple of variables and triplet of variables. For any couple of features X_i, X_j, the estimation of the conditional mutual information given Y requires first an estimation of the model parameter α according to the training data.

The most naive form of multi-variable density would be piece-wise constant, thus discretisation with features of the form $F = 1_{X \geq t}$ where X is one of the original continuous feature. Such a model would lead to the same weakness as those described above for the multi-class situation.

If a more sophisticated model can legitimately be used – for instance multi-dimensional Gaussian – the only difficulty is the computation of the conditional mutual information itself, requiring sums over the space of values of products and ratios of such expressions. Depending on the existence of analytical form of this sum, the algorithm may require numerical integration and heavy computations. Nevertheless, even if the computation of the conditional mutual information is expensive, the lazy evaluation trick presented in §2 can still be used, reducing the cost by the same amount as in the provided results.

8. Conclusion

We have presented a simple and very efficient scheme for feature selection in a context of classification. On the experiments we have done CMIM is the best feature selection method except in one case (SVM for the thrombin experiment). Combined with a naive Bayesian classifier the scores

we obtained are comparable or better than those of state-of-the-art techniques such as boosting or Support Vector Machines, while requiring a training time of a few tenth of a second.

Because of its high speed, this learning method could be used to tune learnt structures on the fly, to adapt them to the specific difficulties of the populations they have to deal with. In the context of face detection, such an on-line training could exploit the specificities of the background population and reduce the false-positive error rate. Also, it could be used in applications requiring the training of a very large number of classifiers. Our current works in object recognition are based on several thousands of classifiers which are built in a few minutes.

References

Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, November 1997.

R. Battiti. Using mutual information for selecting features in supervised neural net learning. In *IEEE Transactions on Neural Networks*, volume 5, 1994.

B. V. Bonnlander and A. S. Weigend. Selecting input variables using mutual information and non-parametric density estimation. In *Proceedings of ISANN*, 1996.

B. E. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *in Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 5:144–152, 1992.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Random forests – random features. Technical Report 567, Department of Statistics, University of California, Berkeley, 1999.

L. Breiman. Some infinity theory for predictors ensembles. Technical Report 579, Department of Statistics, University of California, Berkeley, 2000.

N. Christiani and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *Proceedings of the International Conference on Machine Learning 2001*, pages 74–81, 2001.

R. Duda and P. Hart. *Pattern classification and scene analysis*. John Wiley & Sons, 1973.

F. Fleuret. Binary feature selection with conditional mutual information. Technical Report RR-4941, INRIA, October 2003.

F. Fleuret and D. Geman. Coarse-to-fine visual selection. *International Journal of Computer Vision*, 41(1/2):85–107, 2001.

F. Fleuret and D. Geman. Fast face detection with precise pose estimation. In *Proceedings of ICPR2002*, volume 1, pages 235–238, 2002.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996a.

Y. Freund and R.E. Schapire. Game theory, on-line prediction and boosting. In *Proc. 9th Annu. Conf. on Comput. Learning Theory*, pages 325–332. ACM Press, New York, NY, 1996b.

J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annal of Statistics*, 28:337–407, 2000.

T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10), 2000.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

I. Guyon, S. Gunn, S. Ben Hur, and G. Dror. Result analysis of the NIPS2003 feature selection challenge. In *Proceedings of the NIPS2004 conference*, 2004.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, pages 273–324, 1997.

D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning 1996*, pages 284–292, 1996.

P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *Proceedings of AAAI-92*, pages 223–228, 1992.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, pages 512–518. MIT Press, 2000.

K. Miyahara and M. J. Pazzani. Collaborative filtering with the simple bayesian classifier. In *Pacific Rim International Conference on Artificial Intelligence*, pages 679–689, 2000.

A. Novikoff. On convergence proofs for perceptrons. In *In Symposium on Mathematical Theory of Automata*, pages 615–622, 1962.

C. A. Ratanamahatana and D. Gunopulos. Feature selection for the naive bayesian classifier using decision trees. *Applied Artificial Intelligence*, 17(5-6):475–487, 2003.

G. Ratsch, T. Onoda, and K.-R. Muller. Regularizing AdaBoost. In *Proceedings of NIPS*, pages 564–570, 1998.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1998.

M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *Proceedings of International Conference on Computer Vision 2003*, pages 281–288, 2003.

L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the International Conference on Machine Learning 2003*, pages 856–863, 2003.

Journal of Machine Learning Research 5 (2004) 1557–1595 Submitted 3/2004; Published 12/2004

The Dynamics of AdaBoost: Cyclic Behavior and Convergence of Margins

Cynthia Rudin[*] CRUDIN@PRINCETON.EDU
Ingrid Daubechies INGRID@MATH.PRINCETON.EDU
Program in Applied and Computational Mathematics
Fine Hall
Washington Road
Princeton University
Princeton, NJ 08544-1000, USA

Robert E. Schapire SCHAPIRE@CS.PRINCETON.EDU
Princeton University
Department of Computer Science
35 Olden St.
Princeton, NJ 08544, USA

Editor: Dana Ron

Abstract

In order to study the convergence properties of the AdaBoost algorithm, we reduce AdaBoost to a nonlinear iterated map and study the evolution of its weight vectors. This dynamical systems approach allows us to understand AdaBoost's convergence properties completely in certain cases; for these cases we find stable cycles, allowing us to explicitly solve for AdaBoost's output.

Using this unusual technique, we are able to show that AdaBoost does not always converge to a maximum margin combined classifier, answering an open question. In addition, we show that "non-optimal" AdaBoost (where the weak learning algorithm does not necessarily choose the best weak classifier at each iteration) may fail to converge to a maximum margin classifier, even if "optimal" AdaBoost produces a maximum margin. Also, we show that if AdaBoost cycles, it cycles among "support vectors", i.e., examples that achieve the same smallest margin.

Keywords: boosting, AdaBoost, dynamics, convergence, margins

1. Introduction

Boosting algorithms are currently among the most popular and most successful algorithms for pattern recognition tasks (such as text classification). AdaBoost (Freund and Schapire, 1997) was the first practical boosting algorithm, and due to its success, a number of similar boosting algorithms have since been introduced (see the review paper of Schapire, 2002, for an introduction, or the review paper of Meir and Rätsch, 2003). Boosting algorithms are designed to construct a "strong" classifier using only a training set and a "weak" learning algorithm. A "weak" classifier produced by the weak learning algorithm has a probability of misclassification that is slightly below 50%, i.e., each weak classifier is only required to perform slightly better than a random guess. A "strong"

∗. C. Rudin's present address is New York University / Howard Hughes Medical Institute, 4 Washington Place, Room 809, New York, NY 10003-6603, USA. Her present email is rudin@nyu.edu.

classifier has a much smaller probability of error on test data. Hence, these algorithms "boost" the weak learning algorithm to achieve a stronger classifier. In order to exploit the weak learning algorithm's advantage over random guessing, the data is reweighted (the relative importance of the training examples is changed) before running the weak learning algorithm at each iteration. That is, AdaBoost maintains a distribution (set of weights) over the training examples, and selects a weak classifier from the weak learning algorithm at each iteration. Training examples that were misclassified by the weak classifier at the current iteration then receive higher weights at the following iteration. The end result is a final combined classifier, given by a thresholded linear combination of the weak classifiers.

AdaBoost does not often seem to suffer from overfitting, even after a large number of iterations (Breiman, 1998; Quinlan, 1996). This lack of overfitting has been explained to some extent by the *margin theory* of Schapire, Freund, Bartlett, and Lee (1998). The *margin* of a boosted classifier is a number between -1 and 1, that according to the margin theory, can be thought of as a confidence measure of a classifier's predictive ability, or as a guarantee on the generalization performance. If the margin of a classifier is large, then it tends to perform well on test data. If the margin is small, then the classifier tends not to perform so well. (The margin of a boosted classifier is also called the *minimum margin over training examples*.) Although the empirical success of a boosting algorithm depends on many factors (e.g., the type of data and how noisy it is, the capacity of the weak learning algorithm, the number of boosting iterations, regularization, entire margin distribution over the training examples), the margin theory does provide a reasonable explanation (though not a complete explanation) of AdaBoost's success, both empirically and theoretically.

Since the margin tends to give a strong indication of a classifier's performance in practice, a natural goal is to find classifiers that achieve a maximum margin. Since the AdaBoost algorithm was invented before the margin theory, the algorithm became popular due to its practical success rather than for its theoretical success (its ability to achieve large margins). Since AdaBoost was not specifically designed to maximize the margin, the question remained whether in fact it does actually maximize the margin. The objective function that AdaBoost minimizes (the exponential loss) is not related to the margin in the sense that one can minimize the exponential loss while simultaneously achieving an arbitrarily bad (small) margin. Thus, AdaBoost does not, in fact, optimize a cost function of the margins (see also Wyner, 2002). It was shown analytically that AdaBoost produces *large* margins, namely, Schapire et al. (1998) showed that AdaBoost achieves at least half of the maximum margin, and Rätsch and Warmuth (2002) have recently tightened this bound slightly. However, because AdaBoost does not necessarily make progress towards increasing the margin at each iteration, the usual techniques for analyzing coordinate algorithms do not apply; for all the extensive theoretical and empirical study of AdaBoost prior to the present work, it remained unknown whether or not AdaBoost always achieves a maximum margin solution.

A number of other boosting algorithms emerged over the past few years that aim more explicitly to maximize the margin at each iteration, such as AdaBoost* (Rätsch and Warmuth, 2002), arc-gv (Breiman, 1999), Coordinate Ascent Boosting and Approximate Coordinate Ascent Boosting (Rudin et al., 2004c,b,a; Rudin, 2004), the linear programming (LP) boosting algorithms including LP-AdaBoost (Grove and Schuurmans, 1998) and LPBoost (Demiriz et al., 2002). (Also see the ε-boosting literature, for example, Rosset et al., 2004.) However, AdaBoost is still used in practice, because it often empirically seems to produce maximum margin classifiers with low generalization error. In fact, under tightly controlled tests, it was shown empirically that the maximum margin algorithms arc-gv and LP-AdaBoost tend to perform *worse* than AdaBoost (Breiman, 1999; Grove

and Schuurmans, 1998). In the experiments of Grove and Schuurmans (1998), AdaBoost achieved margins that were almost as large, (but not quite as large) as those of the LP algorithms when stopped after a large number of iterations, yet often achieved lower generalization error. AdaBoost is also easy to program, and in our trials, it seems to converge the fastest (with respect to the margin) among the coordinate-based boosting algorithms.

Another surprising result of empirical trials is that AdaBoost does seem to be converging to maximum margin solutions *asymptotically* in the numerical experiments of Grove and Schuurmans (1998) and Rätsch and Warmuth (2002). Grove and Schuurmans have questioned whether AdaBoost is simply a "general, albeit very slow, LP solver". If AdaBoost is simply a margin-maximization algorithm, then why are other algorithms that achieve the same margin performing worse than AdaBoost? Is AdaBoost simply a fancy margin-maximization algorithm in disguise, or is it something more? As we will see, the answers are sometimes yes and sometimes no. So clearly the margins do not tell the whole story.

AdaBoost, as shown repeatedly (Breiman, 1997; Friedman et al., 2000; Rätsch et al., 2001; Duffy and Helmbold, 1999; Mason et al., 2000), is actually a coordinate descent algorithm on a particular exponential loss function. However, minimizing this function in other ways does not necessarily achieve large margins; the process of coordinate descent must be somehow responsible. Hence, we look to AdaBoost's dynamics to understand the process by which the margin is generated.

In this work, we took an unusual approach to this problem. We simplified AdaBoost to reveal a nonlinear iterated map for AdaBoost's weight vector. This iterated map gives a direct relation between the weights at time t and the weights at time $t+1$, including renormalization, and thus provides a much more concise mapping than the original algorithm. We then analyzed this dynamical system in specific cases. Using a small toolbox of techniques for analyzing dynamical systems, we were able to avoid the problem that progress (with respect to the margin) does not occur at every iteration. Instead, we measure progress another way; namely, via the convergence towards limit cycles.

To explain this way of measuring progress more clearly, we have found that for some specific cases, the weight vector of AdaBoost produces limit cycles that can be analytically stated, and are stable. When stable limit cycles exist, the convergence of AdaBoost can be understood. Thus, we are able to provide the key to answering the question of AdaBoost's convergence to maximum margin solutions: a collection of examples in which AdaBoost's convergence can be completely understood.

Using a very low-dimensional example (8×8, i.e., 8 weak classifiers and 8 training examples), we are able to show that AdaBoost does not always produce a maximum margin solution, finally answering the open question.

There are two interesting cases governing the dynamics of AdaBoost: the case where the optimal weak classifiers are chosen at each iteration (the "optimal" case), and the case where permissible non-optimal weak classifiers may be chosen (the "non-optimal" case). In the optimal case (which is the case we usually consider), the weak learning algorithm is required to choose a weak classifier that has the largest edge at every iteration, where the edge measures the performance of the weak learning algorithm. In the non-optimal case, the weak learning algorithm may choose any weak classifier as long as its edge exceeds ρ, the maximum margin achievable by a combined classifier. This is a natural notion of non-optimality for boosting, thus it provides a natural sense in which to measure robustness. Based on large scale experiments and a gap in theoretical bounds, Rätsch and Warmuth (2002) conjectured that AdaBoost does not necessarily converge to a maximum margin

classifier in the non-optimal case, i.e., that AdaBoost is not robust in this sense. In practice, the weak classifiers are generated by CART or another weak learning algorithm, implying that the choice need not always be optimal.

In Section 8, we show this conjecture to be true using a 4×5 example. That is, we show that "non-optimal AdaBoost" (AdaBoost in the non-optimal case) may not converge to a maximum margin solution, even in cases where "optimal AdaBoost" does.

Empirically, we have found very interesting and remarkable cyclic dynamics in many different low-dimensional cases (many more cases than the ones analyzed in this paper), for example, those illustrated in Figure 6. In fact, we have empirically found that AdaBoost produces cycles on randomly generated matrices – even on random matrices with hundreds of dimensions. On low-dimensional random matrices, cycles are almost always produced in our experiments. Thus, the story of AdaBoost's dynamics does not end with the margins; it is important to study AdaBoost's dynamics in more general cases where these cycles occur in order to understand its convergence properties.

To this extent, we prove that if AdaBoost cycles, it cycles only among a set of "support vectors" that achieve the same smallest margin among training examples. In this sense, we confirm observations of Caprile et al. (2002) who previously studied the dynamical behavior of boosting, and who also identified two sorts of examples which they termed "easy" and "hard." In addition, we give sufficient conditions for AdaBoost to achieve a maximum margin solution when cycling occurs. We also show that AdaBoost treats identically classified examples as one example, in the sense we will describe in Section 6. In Section 10, we discuss a case in which AdaBoost exhibits indications of chaotic behavior, namely sensitivity to initial conditions, and movement into and out of cyclic behavior.

We proceed as follows. In Section 2 we introduce some notation and state the AdaBoost algorithm. Then in Section 3 we decouple the dynamics for AdaBoost in the binary case so that we have a nonlinear iterated map. In Section 4, we analyze these dynamics for a simple case: the case where each weak classifier has one misclassified training example. In a 3×3 example, we find that the weight vectors always converge to one of two stable limit cycles, allowing us to calculate AdaBoost's output vector directly. From this, we can prove the output of AdaBoost yields the best possible margin. We generalize this case to $m \times m$ in Section 5. In Section 6 we discuss identically classified examples. Namely, we show that the weights on identically classified training examples can be shifted among these examples while preserving the cycle; that is, manifolds of stable cycles can occur. For an extension of the simple 3×3 case, we show that manifolds of cycles exist and are stable. In Section 7 we show that the training examples AdaBoost cycles upon are "support vectors" in that they all achieve the same margin. In the process, we provide a formula to directly calculate the margin from the cycle parameters. We also give sufficient conditions for AdaBoost to produce a maximum margin classifier when cycling occurs. Then in Section 8 we produce an example to show non-robustness of AdaBoost in the non-optimal case. In Section 9, we produce the example discussed above to show that AdaBoost may not converge to a maximum margin solution. And finally in Section 10, we provide a case for which AdaBoost exhibits indications of chaotic behavior.

2. Notation and Introduction to AdaBoost

The training set consists of examples with labels $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$, where $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \{-1, 1\}$. The space $\mathcal{X}$ never appears explicitly in our calculations. Let $\mathcal{H} = \{h_1, \ldots, h_n\}$ be the set of all possible weak classifiers that can be produced by the weak learning algorithm, where $h_j : \mathcal{X} \to \{1, -1\}$. We assume that if h_j appears in $\mathcal{H}$, then $-h_j$ also appears in $\mathcal{H}$. Since our classifiers are binary, and since we restrict our attention to their behavior on a finite training set, we can assume the number of weak classifiers n is finite. We typically think of n as being very large, $m \ll n$, which makes a gradient descent calculation impractical because n, the number of dimensions, is too large; hence, AdaBoost uses coordinate descent instead, where only one weak classifier is chosen at each iteration.

We define an $m \times n$ matrix $\mathbf{M}$ where $M_{ij} = y_i h_j(\mathbf{x}_i)$, i.e., $M_{ij} = +1$ if training example i is classified correctly by weak classifier h_j, and -1 otherwise. We assume that no column of $\mathbf{M}$ has all +1's, that is, no weak classifier can classify all the training examples correctly. (Otherwise the learning problem is trivial. In this case, AdaBoost will have an undefined step size.) Although $\mathbf{M}$ is too large to be explicitly constructed in practice, mathematically, it acts as the only "input" to AdaBoost in this notation, containing all the necessary information about the weak learning algorithm and training examples.

AdaBoost computes a set of coefficients over the weak classifiers. At iteration t, the (unnormalized) coefficient vector is denoted $\boldsymbol{\lambda}_t$; i.e., the coefficient of weak classifier h_j determined by AdaBoost at iteration t is $\lambda_{t,j}$. The final combined classifier that AdaBoost outputs is $f_{\boldsymbol{\lambda}_{tmax}}$ given via $\boldsymbol{\lambda}_{t_{max}} / \|\boldsymbol{\lambda}_{t_{max}}\|_1$:

$$f_{\boldsymbol{\lambda}} = \frac{\sum_{j=1}^{n} \lambda_j h_j}{\|\boldsymbol{\lambda}\|_1} \quad \text{where} \quad \|\boldsymbol{\lambda}\|_1 = \sum_{j=1}^{n} |\lambda_j|.$$

In the specific examples we provide, either h_j or $-h_j$ remains unused over the course of AdaBoost's iterations, so all values of $\lambda_{t,j}$ are non-negative. The *margin of training example i* is defined by $y_i f_{\boldsymbol{\lambda}}(\mathbf{x}_i)$. Informally, one can think of the margin of a training example as the distance (by some measure) from the example to the decision boundary, $\{\mathbf{x} : f_{\boldsymbol{\lambda}}(\mathbf{x}) = 0\}$.

A boosting algorithm maintains a distribution, or set of weights, over the training examples that is updated at each iteration t. This distribution is denoted $\mathbf{d}_t \in \Delta_m$, and $\mathbf{d}_t^T$ is its transpose. Here, Δ_m denotes the simplex of m-dimensional vectors with non-negative entries that sum to 1. At each iteration t, a weak classifier h_{j_t} is selected by the weak learning algorithm. The *probability of error* at iteration t, denoted d_-, for the selected weak classifier h_{j_t} on the training examples (weighted by $\mathbf{d}_t$) is $\sum_{\{i:M_{ij_t}=-1\}} d_{t,i}$. Also, denote $d_+ := 1 - d_-$. Note that d_+ and d_- depend on t; although we have simplified the notation, the iteration number will be clear from the context. The *edge* of weak classifier j_t at time t with respect to the training examples is $(\mathbf{d}_t^T \mathbf{M})_{j_t}$, which can be written as

$$(\mathbf{d}_t^T \mathbf{M})_{j_t} = \sum_{i:M_{ij_t}=1} d_{t,i} - \sum_{i:M_{ij_t}=-1} d_{t,i} = d_+ - d_- = 1 - 2d_-.$$

Thus, a smaller edge indicates a higher probability of error. For the *optimal* case (the case we usually consider), we will require the weak learning algorithm to give us the weak classifier with the largest possible edge at each iteration,

$$j_t \in \operatorname*{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j,$$

i.e., j_t is the weak classifier that performs the best on the training examples weighted by $\mathbf{d}_t$. For the *non-optimal* case (which we consider in Section 8), we only require a weak classifier whose edge exceeds ρ, where ρ is the largest possible margin that can be attained for $\mathbf{M}$, i.e.,

$$j_t \in \{j : (\mathbf{d}_t^T \mathbf{M})_j \geq \rho\}.$$

(The value ρ is defined formally below.) The edge for the chosen weak classifier j_t at iteration t is denoted r_t, i.e., $r_t = (\mathbf{d}_t^T \mathbf{M})_{j_t}$. Note that $d_+ = (1+r_t)/2$ and $d_- = (1-r_t)/2$.

The margin theory developed via a set of generalization bounds that are based on the margin distribution of the training examples (Schapire et al., 1998; Koltchinskii and Panchenko, 2002). These bounds can be reformulated (in a slightly weaker form) in terms of the minimum margin, which was the focus of previous work by Breiman (1999), Grove and Schuurmans (1998), and Rätsch and Warmuth (2002). Thus, these bounds suggest maximizing the minimum margin over training examples to achieve a low probability of error over test data. Hence, our goal is to find a normalized vector $\tilde{\boldsymbol{\lambda}} \in \Delta_n$ that maximizes the minimum margin over training examples, $\min_i (\mathbf{M}\tilde{\boldsymbol{\lambda}})_i$ (or equivalently $\min_i y_i f_{\boldsymbol{\lambda}}(\mathbf{x}_i)$). That is, we wish to find a vector

$$\tilde{\boldsymbol{\lambda}} \in \operatorname*{argmax}_{\boldsymbol{\lambda} \in \Delta_n} \min_i (\mathbf{M}\bar{\boldsymbol{\lambda}})_i.$$

We call this minimum margin over training examples (i.e., $\min_i (\mathbf{M}\boldsymbol{\lambda})_i / \|\boldsymbol{\lambda}\|_1$) the ℓ_1-*margin* or simply *margin* of classifier $\boldsymbol{\lambda}$. Any training example that achieves this minimum margin will be called a *support vector*. Due to the von Neumann Min-Max Theorem for 2-player zero-sum games,

$$\min_{\mathbf{d} \in \Delta_m} \max_j (\mathbf{d}^T \mathbf{M})_j = \max_{\tilde{\boldsymbol{\lambda}} \in \Delta_n} \min_i (\mathbf{M}\tilde{\boldsymbol{\lambda}})_i.$$

That is, the minimum value of the edge (left hand side) corresponds to the maximum value of the margin (i.e., the maximum value of the minimum margin over training examples, right hand side). We denote this value by ρ. One can think of ρ as measuring the worst performance of the best combined classifier, $\min_i (\mathbf{M}\tilde{\boldsymbol{\lambda}})_i$.

The "unrealizable" or "non-separable" case where $\rho = 0$ is fully understood (Collins et al., 2002). For this work, we assume $\rho > 0$ and study the less understood "realizeable" or "separable" case. In both the non-separable and separable cases, AdaBoost converges to a minimizer of the empirical loss function

$$F(\boldsymbol{\lambda}) := \sum_{i=1}^{m} e^{-(\mathbf{M}\boldsymbol{\lambda})_i}.$$

In the non-separable case, the $\mathbf{d}_t$'s converge to a fixed vector (Collins et al., 2002). In the separable case, the $\mathbf{d}_t$'s cannot converge to a fixed vector, and the minimum value of F is 0, occurring as $||\boldsymbol{\lambda}||_1 \to \infty$. It is important to appreciate that this tells us nothing about the value of the margin achieved by AdaBoost or any other procedure designed to minimize F. To see why, consider any $\bar{\boldsymbol{\lambda}} \in \Delta_n$ such that $(\mathbf{M}\bar{\boldsymbol{\lambda}})_i > 0$ for all i (assuming we are in the separable case so such a $\bar{\boldsymbol{\lambda}}$ exists). Then $\lim_{a \to \infty} a\bar{\boldsymbol{\lambda}}$ will produce a minimum value for F, but the original normalized $\bar{\boldsymbol{\lambda}}$ need not yield a maximum margin. To clarify, any normalized $\bar{\boldsymbol{\lambda}}$ for which $(\mathbf{M}\bar{\boldsymbol{\lambda}})_i > 0$ for all i produces a classifier that classifies all training examples correctly, has unnormalized counterparts that attain values of F arbitrarily close to 0, yet may produce a classifier with arbitrarily *small* margin. In other words, an arbitrary algorithm that minimizes F can achieve an arbitrarily bad margin. So it must be the *process*

AdaBoost ("optimal" case):

1. **Input:** Matrix $\mathbf{M}$, No. of iterations t_{max}
2. **Initialize:** $\lambda_{1,j} = 0$ for $j = 1,...,n$
3. **Loop for** $t = 1,...,t_{max}$
   (a) $d_{t,i} = e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_i} / \sum_{\tilde{i}=1}^{m} e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{\tilde{i}}}$ for $i = 1,...,m$
   (b) $j_t \in \operatorname{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j$
   (c) $r_t = (\mathbf{d}_t^T \mathbf{M})_{j_t}$
   (d) $\alpha_t = \frac{1}{2} \ln\left(\frac{1+r_t}{1-r_t}\right)$
   (e) $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \alpha_t \mathbf{e}_{j_t}$, where $\mathbf{e}_{j_t}$ is 1 in position j_t and 0 elsewhere.
4. **Output:** $\boldsymbol{\lambda}_{t_{max}} / \|\boldsymbol{\lambda}_{t_{max}}\|_1$

Figure 1: Pseudocode for the AdaBoost algorithm.

of coordinate descent that awards AdaBoost its ability to increase margins, not simply AdaBoost's ability to minimize F. The value of the function F tells us very little about the value of the margin; even asymptotically, it only tells us whether the margin is positive or not.

Figure 1 shows pseudocode for the AdaBoost algorithm. Usually the $\boldsymbol{\lambda}_1$ vector is initialized to zero, so that all the training examples are weighted equally during the first iteration. The weight vector $\mathbf{d}_t$ is adjusted so that training examples that were misclassified at the previous iteration are weighted more highly, so they are more likely to be correctly classified at the next iteration. The weight vector $\mathbf{d}_t$ is determined from the vector of coefficients $\boldsymbol{\lambda}_t$, which has been updated. The map from $\mathbf{d}_t$ to $\mathbf{d}_{t+1}$ also involves renormalization, so it is not a very direct map when written in this form. Thus on each round of boosting, the distribution $\mathbf{d}_t$ is updated and renormalized (Step 3a), classifier j_t with maximum edge (minimum probability of error) is selected (Step 3b), and the weight of that classifier is updated (Step 3e). Note that $\lambda_{t,j} = \sum_{\tilde{t}=1}^{t} \alpha_{\tilde{t}} \mathbf{1}_{j_{\tilde{t}}=j}$ where $\mathbf{1}_{j_{\tilde{t}}=j}$ is 1 if $j_{\tilde{t}} = j$ and 0 otherwise.

3. The Iterated Map Defined By AdaBoost

AdaBoost can be reduced to an iterated map for the $\mathbf{d}_t$'s, as shown in Figure 2. This map gives a direct relationship between $\mathbf{d}_t$ and $\mathbf{d}_{t+1}$, taking the normalization of Step 3a into account automatically. For the cases considered in Sections 4, 5, and 6, we only need to understand the dynamics of Figure 2 in order to compute the final coefficient vector that AdaBoost will output. Initialize $d_{1,i} = 1/m$ for $i = 1,...,m$ as in the first iteration of AdaBoost. Also recall that all values of r_t are nonnegative since $r_t \geq \rho > 0$.

To show the equivalence with AdaBoost, consider the iteration defined by AdaBoost and reduce as follows. Since:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+r_t}{1-r_t}\right), \text{ we have } e^{-(M_{ij_t}\alpha_t)} = \left(\frac{1-r_t}{1+r_t}\right)^{\frac{1}{2}M_{ij_t}} = \left(\frac{1-M_{ij_t}r_t}{1+M_{ij_t}r_t}\right)^{\frac{1}{2}}.$$

Iterated Map Defined by AdaBoost

1. $j_t \in \operatorname{argmax}_j (\mathbf{d}_t^T \mathbf{M})_j$
2. $r_t = (\mathbf{d}_t^T \mathbf{M})_{j_t}$
3. $d_{t+1,i} = \frac{d_{t,i}}{1+M_{ij_t} r_t}$ for $i = 1, \ldots, m$

Figure 2: The nonlinear iterated map obeyed by AdaBoost's weight vectors. This dynamical system provides a direct map from $\mathbf{d}_t$ to $\mathbf{d}_{t+1}$.

Here, we have used the fact that $\mathbf{M}$ is a binary matrix. The iteration defined by AdaBoost combined with the equation above yields:

$$d_{t+1,i} = \frac{e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_i} e^{-(M_{ij_t}\alpha_t)}}{\sum_{\bar{i}=1}^m e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{\bar{i}}} e^{-(M_{\bar{i}j_t}\alpha_t)}} = \frac{d_{t,i}}{\sum_{\bar{i}=1}^m d_{t,\bar{i}} \left(\frac{1-M_{\bar{i}j_t} r_t}{1+M_{\bar{i}j_t} r_t}\right)^{\frac{1}{2}} \left(\frac{1+M_{ij_t} r_t}{1-M_{ij_t} r_t}\right)^{\frac{1}{2}}}.$$

Here, we have divided numerator and denominator by $\sum_{\bar{i}=1}^m e^{-(\mathbf{M}\boldsymbol{\lambda}_t)_{\bar{i}}}$. For each i such that $M_{ij_t} = 1$, we find:

$$\begin{aligned} d_{t+1,i} &= \frac{d_{t,i}}{\sum_{\{\bar{i}:M_{\bar{i}j_t}=1\}} d_{t,\bar{i}} \left(\frac{1-r_t}{1+r_t}\right)^{\frac{1}{2}} \left(\frac{1+r_t}{1-r_t}\right)^{\frac{1}{2}} + \sum_{\{\bar{i}:M_{\bar{i}j_t}=-1\}} d_{t,\bar{i}} \left(\frac{1+r_t}{1-r_t}\right)^{\frac{1}{2}} \left(\frac{1+r_t}{1-r_t}\right)^{\frac{1}{2}}} \\ &= \frac{d_{t,i}}{d_+ + d_- \left(\frac{1+r_t}{1-r_t}\right)} = \frac{d_{t,i}}{\frac{1+r_t}{2} + \frac{1-r_t}{2} \left(\frac{1+r_t}{1-r_t}\right)} = \frac{d_{t,i}}{1+r_t}. \end{aligned}$$

Likewise, for each i such that $M_{ij_t} = -1$, we find $d_{t+1,i} = \frac{d_{t,i}}{1-r_t}$. Thus our reduction is complete. To check that $\sum_{i=1}^m d_{t+1,i} = 1$, i.e., that renormalization has been taken into account by the iterated map, we calculate:

$$\sum_{i=1}^m d_{t+1,i} = \frac{1}{1+r_t} d_+ + \frac{1}{1-r_t} d_- = \frac{(1+r_t)}{2(1+r_t)} + \frac{(1-r_t)}{2(1-r_t)} = 1.$$

For the iterated map, fixed points (rather than cycles or other dynamics) occur when the training data fails to be separable by the set of weak classifiers. In that case, the analysis of Collins, Schapire, and Singer (2002) shows that the iterated map will converge to a fixed point, and that the $\boldsymbol{\lambda}_t's$ will asymptotically attain the minimum value of the convex function $F(\boldsymbol{\lambda}) := \sum_{i=1}^m e^{-(\mathbf{M}\boldsymbol{\lambda})_i}$, which is strictly positive in the non-separable case. Consider the possibility of fixed points for the $\mathbf{d}_t$'s in the separable case $\rho > 0$. From our dynamics, we can see that this is not possible, since $r_t \geq \rho > 0$ and for any i such that $d_{t,i} > 0$,

$$d_{t+1,i} = \frac{d_{t,i}}{(1+M_{i,j_t} r_t)} \neq d_{t,i}.$$

Thus, we have shown that AdaBoost does not produce fixed points in the separable case.

4. The Dynamics of AdaBoost in the Simplest Case : The 3×3 Case

In this section, we will introduce a simple 3×3 input matrix (in fact, the simplest non-trivial matrix) and analyze the convergence of AdaBoost in this case, using the iterated map of Section 3. We will show that AdaBoost does produce a maximum margin solution, remarkably through convergence to one of two stable limit cycles. We extend this example to the $m \times m$ case in Section 5, where AdaBoost produces at least $(m-1)!$ stable limit cycles, each corresponding to a maximum margin solution. We will also extend this example in Section 6 to include manifolds of cycles.

Consider the input matrix

$$\mathbf{M} = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$$

corresponding to the case where each classifier misclassifies one of three training examples. We could add columns to include the negated version of each weak classifier, but those columns would never be chosen by AdaBoost, so they have been removed for simplicity. The value of the margin for the best combined classifier defined by $\mathbf{M}$ is $1/3$. How will AdaBoost achieve this result? We will proceed step by step.

Assume we are in the optimal case, where $j_t \in \text{argmax}_j(\mathbf{d}_t^T\mathbf{M})_j$. Consider the dynamical system on the simplex Δ_3 defined by our iterated map in Section 3. In the triangular region with vertices $(0,0,1),(\frac{1}{3},\frac{1}{3},\frac{1}{3}),(0,1,0)$, j_t will be 1 for Step 1 of the iterated map. That is, within this region, $d_{t,1} < d_{t,2}$ and $d_{t,1} < d_{t,3}$, so j_t will be 1. Similarly, we have regions for $j_t = 2$ and $j_t = 3$ (see Figure 3(a)).

AdaBoost was designed to set the edge of the previous weak classifier to 0 at each iteration, that is, $\mathbf{d}_{t+1}$ will always satisfy $(\mathbf{d}_{t+1}^T\mathbf{M})_{j_t} = 0$. To see this using the iterated map,

$$\begin{aligned}(\mathbf{d}_{t+1}^T\mathbf{M})_{j_t} &= \sum_{\{i:M_{ij_t}=1\}} d_{t,i}\frac{1}{1+r_t} - \sum_{\{i:M_{ij_t}=-1\}} d_{t,i}\frac{1}{1-r_t} \\ &= d_+\frac{1}{1+r_t} - d_-\frac{1}{1-r_t} = \frac{1+r_t}{2}\frac{1}{1+r_t} - \frac{1-r_t}{2}\frac{1}{1-r_t} = 0. \end{aligned} \quad (1)$$

This implies that after the first iteration, the $\mathbf{d}_t$'s are restricted to

$$\{\mathbf{d} : [(\mathbf{d}^T\mathbf{M})_1 = 0] \bigcup [(\mathbf{d}^T\mathbf{M})_2 = 0] \bigcup [(\mathbf{d}^T\mathbf{M})_3 = 0]\}.$$

Thus, it is sufficient for our dynamical system to be analyzed on the edges of a triangle with vertices $\left(0,\frac{1}{2},\frac{1}{2}\right)$, $\left(\frac{1}{2},0,\frac{1}{2}\right)$, $\left(\frac{1}{2},\frac{1}{2},0\right)$ (see Figure 3(b)). That is, within one iteration, the 2-dimensional map on the simplex Δ_3 reduces to a 1-dimensional map on the edges of the triangle.

Consider the possibility of periodic cycles for the $\mathbf{d}_t$'s. If there are periodic cycles of length T, then the following condition must hold for $\mathbf{d}_1^{cyc},...,\mathbf{d}_T^{cyc}$ in the cycle: For each i, either

- $d_{1,i}^{cyc} = 0$, or
- $\prod_{t=1}^{T}(1 + M_{ij_t}r_t^{cyc}) = 1$,

where $r_t^{cyc} = (\mathbf{d}_t^{cycT}\mathbf{M})_{j_t}$. (As usual, $\mathbf{d}_t^{cycT} := (\mathbf{d}_t^{cyc})^T$, superscript T denotes transpose.) The statement above follows directly from the reduced map iterated T times. In fact, the first condition $d_{1,i}^{cyc} = 0$ implies $d_{t,i}^{cyc} = 0$ for all $t \in \{1,...,T\}$. We call the second condition the *cycle condition*.

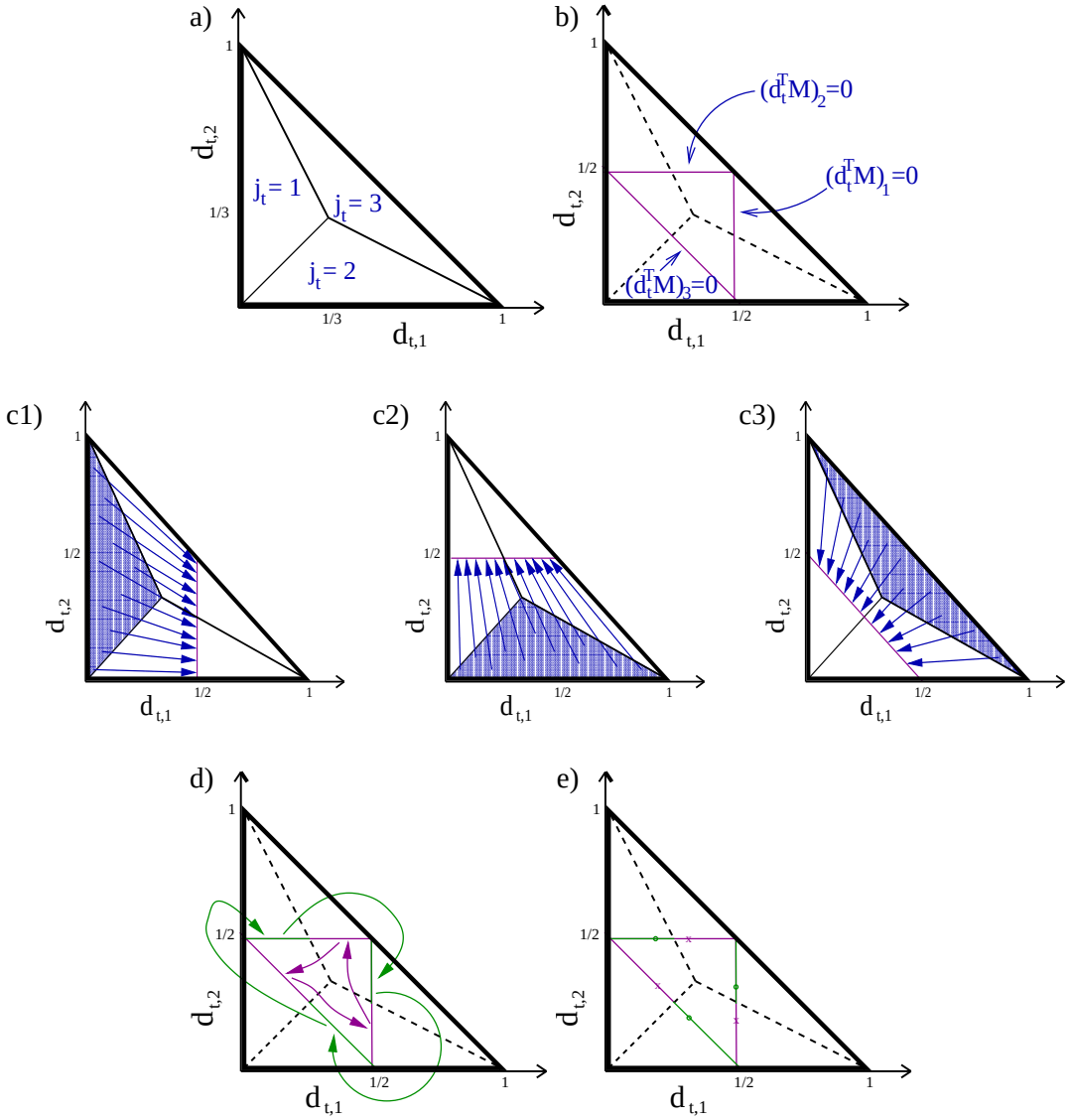


Figure 3: (a) Regions of $\mathbf{d}_t$-space where classifiers $j_t = 1, 2, 3$ will respectively be selected for Step 1 of the iterated map of Figure 2. Since $d_{t,3} = 1 - d_{t,2} - d_{t,1}$, this projection onto the first two coordinates $d_{t,1}$ and $d_{t,2}$ completely characterizes the map. (b) Regardless of the initial position $\mathbf{d}_1$, the weight vectors at all subsequent iterations $\mathbf{d}_2, \ldots, \mathbf{d}_{t_{max}}$ will be restricted to lie on the edges of the inner triangle which is labelled. (c1) Within one iteration, the triangular region where $j_t = 1$ maps to the line $\{\mathbf{d} : (\mathbf{d}^T \mathbf{M})_1 = 0\}$. The arrows indicate where various points in the shaded region will map at the following iteration. The other two regions have analogous dynamics as shown in (c2) and (c3). (d) There are six total subregions of the inner triangle (two for each of the three edges). Each subregion is mapped to the interior of another subregion as indicated by the arrows. (e) Coordinates for the two 3-cycles. The approximate positions $\mathbf{d}_1^{cyc}$, $\mathbf{d}_2^{cyc}$, and $\mathbf{d}_3^{cyc}$ for one of the 3-cycles are denoted by a small 'o', the positions for the other cycle are denoted by a small 'x'.

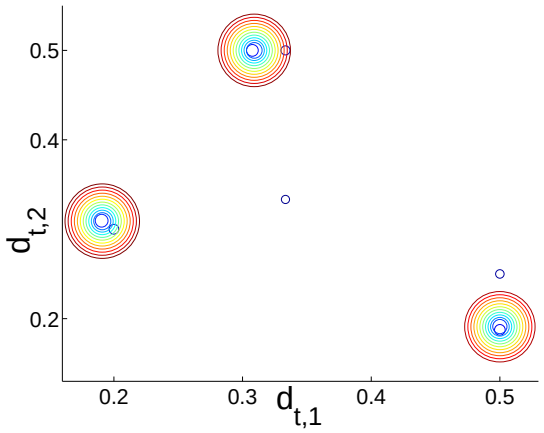


Figure 4: 50 iterations of AdaBoost showing convergence of $\mathbf{d}_t$'s to a cycle. Small rings indicate earlier iterations of AdaBoost, while larger rings indicate later iterations. There are many concentric rings at positions $\mathbf{d}_1^{cyc}$, $\mathbf{d}_2^{cyc}$, and $\mathbf{d}_3^{cyc}$.

Consider the possibility of a periodic cycle of length 3, cycling through each weak classifier once. For now, assume $j_1 = 1, j_2 = 2, j_3 = 3$, but without loss of generality one can choose $j_1 = 1, j_2 = 3, j_3 = 2$, which yields another cycle. Assume $d_{1,i}^{cyc} > 0$ for all i. From the cycle condition,

$$\begin{aligned}
1 &= (1+M_{ij_1}r_1^{cyc})(1+M_{ij_2}r_2^{cyc})(1+M_{ij_3}r_3^{cyc}) \quad \text{for } i=1,2, \text{ and } 3, \text{ i.e.,} \\
1 &= (1-r_1^{cyc})(1+r_2^{cyc})(1+r_3^{cyc}) \quad \text{for } i=1, && (2)\\
1 &= (1+r_1^{cyc})(1-r_2^{cyc})(1+r_3^{cyc}) \quad \text{for } i=2, && (3)\\
1 &= (1+r_1^{cyc})(1+r_2^{cyc})(1-r_3^{cyc}) \quad \text{for } i=3. && (4)
\end{aligned}$$

From (2) and (3),

$$(1-r_1^{cyc})(1+r_2^{cyc}) = (1+r_1^{cyc})(1-r_2^{cyc}),$$

thus $r_1^{cyc} = r_2^{cyc}$. Similarly, $r_2^{cyc} = r_3^{cyc}$ from (3) and (4), so $r_1^{cyc} = r_2^{cyc} = r_3^{cyc}$. Using either (2), (3), or (4) to solve for $r := r_1^{cyc} = r_2^{cyc} = r_3^{cyc}$ (taking positive roots since $r > 0$), we find the value of the edge for every iteration in the cycle to be equal to the golden ratio minus one, i.e.,

$$r = \frac{\sqrt{5}-1}{2}.$$

Now, let us solve for the weight vectors in the cycle, $\mathbf{d}_1^{cyc}$, $\mathbf{d}_2^{cyc}$, and $\mathbf{d}_3^{cyc}$. At $t = 2$, the edge with respect to classifier 1 is 0. Again, it is required that each $\mathbf{d}_t^{cyc}$ lies on the simplex Δ_3.

$$\begin{aligned}
(\mathbf{d}_2^{cyc\,T}\mathbf{M})_1 = 0 \quad &\text{and} \quad \sum_{i=1}^{3} d_{2,i}^{cyc} = 1, \quad \text{that is,} \\
-d_{2,1}^{cyc} + d_{2,2}^{cyc} + d_{2,3}^{cyc} = 0 \quad &\text{and} \quad d_{2,1}^{cyc} + d_{2,2}^{cyc} + d_{2,3}^{cyc} = 1, \\
\text{thus,} \quad d_{2,1}^{cyc} \quad &= \quad \frac{1}{2}.
\end{aligned}$$

Since $d^{cyc}_{2,1} = \frac{1}{2}$, we have $d^{cyc}_{2,2} = \frac{1}{2} - d^{cyc}_{2,3}$. At $t = 3$, the edge with respect to classifier 2 is 0. From the iterated map, we can write $\mathbf{d}^{cyc}_3$ in terms of $\mathbf{d}^{cyc}_2$.

$$0 = (\mathbf{d}^{cyc\,T}_3 \mathbf{M})_2 = \sum_{i=1}^{3} \frac{M_{i2} d^{cyc}_{2,i}}{1 + M_{i2} r} = \frac{\frac{1}{2}}{1+r} - \frac{\frac{1}{2} - d^{cyc}_{2,3}}{1-r} + \frac{d^{cyc}_{2,3}}{1+r}, \quad \text{so}$$

$$d^{cyc}_{2,3} = \frac{r}{2} = \frac{\sqrt{5}-1}{4} \quad \text{and thus} \quad d^{cyc}_{2,2} = \frac{1}{2} - d^{cyc}_{2,3} = \frac{3-\sqrt{5}}{4}.$$

Now that we have found $\mathbf{d}^{cyc}_2$, we can recover the rest of the cycle:

$$\begin{aligned}
\mathbf{d}^{cyc}_1 &= \left(\frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4}, \frac{1}{2} \right)^T, \\
\mathbf{d}^{cyc}_2 &= \left(\frac{1}{2}, \frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4} \right)^T, \\
\mathbf{d}^{cyc}_3 &= \left(\frac{\sqrt{5}-1}{4}, \frac{1}{2}, \frac{3-\sqrt{5}}{4} \right)^T.
\end{aligned}$$

To check that this actually is a cycle, starting from $\mathbf{d}^{cyc}_1$, AdaBoost will choose $j_t = 1$. Then $r_1 = (\mathbf{d}^{cyc\,T}_1 \mathbf{M})_1 = \frac{\sqrt{5}-1}{2}$. Now, computing $\frac{d^{cyc}_{1,i}}{1+M_{i,1} r_1}$ for all i yields $\mathbf{d}^{cyc}_2$. In this way, AdaBoost will cycle between weak classifiers $j = 1, 2, 3, 1, 2, 3$, etc.

The other 3-cycle can be determined similarly:

$$\begin{aligned}
\mathbf{d}^{cyc'}_1 &= \left(\frac{3-\sqrt{5}}{4}, \frac{1}{2}, \frac{\sqrt{5}-1}{4} \right)^T, \\
\mathbf{d}^{cyc'}_2 &= \left(\frac{1}{2}, \frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4} \right)^T, \\
\mathbf{d}^{cyc'}_3 &= \left(\frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4}, \frac{1}{2} \right)^T.
\end{aligned}$$

Since we always start from the initial condition $\mathbf{d}_1 = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)^T$, the initial choice of j_t is arbitrary; all three weak classifiers are within the argmax set in Step 1 of the iterated map. This arbitrary step, along with another arbitrary choice at the second iteration, determines which of the two cycles the algorithm will choose; as we will see, the algorithm must converge to one of these two cycles.

To show that these cycles are globally stable, we will show that the map is a contraction from each subregion of the inner triangle into another subregion. We only need to consider the one-dimensional map defined on the edges of the inner triangle, since within one iteration, every trajectory starting within the simplex Δ_3 lands somewhere on the edges of the inner triangle. The edges of the inner triangle consist of 6 subregions, as shown in Figure 3(d). We will consider one subregion, the segment from $\left(0, \frac{1}{2}, \frac{1}{2}\right)^T$ to $\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)^T$, or simply $\left(x, \frac{1}{2}, \frac{1}{2} - x\right)^T$ where $x \in (0, \frac{1}{4})$. (We choose not to deal with the endpoints since we will show they are unstable; thus the dynamics never reach or

converge to these points. For the first endpoint the map is not defined, and for the second, the map is ambiguous; not well-defined.) For this subregion $j_t = 1$, and the next iterate is

$$\left(\frac{x}{1-(1-2x)}, \frac{\frac{1}{2}}{1+(1-2x)}, \frac{\frac{1}{2}-x}{1+(1-2x)}\right)^T = \left(\frac{1}{2}, \frac{1}{4(1-x)}, \frac{1}{2} - \frac{1}{4(1-x)}\right)^T.$$

To compare the length of the new interval with the length of the previous interval, we use the fact that there is only one degree of freedom. A position on the previous interval can be uniquely determined by its first component $x \in (0, \frac{1}{4})$. A position on the new interval can be uniquely determined by its second component taking values $\frac{1}{4(1-x)}$, where we still have $x \in (0, \frac{1}{4})$. The map

$$x \mapsto \frac{1}{4(1-x)}$$

is a contraction. To see this, the slope of the map is $\frac{1}{4(1-x)^2}$, taking values within the interval $(\frac{1}{4}, \frac{4}{9})$. Thus the map is continuous and monotonic, with absolute slope strictly less than 1. The next iterate will appear within the interval $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)^T$ to $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right)^T$, which is strictly contained within the subregion connecting $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)^T$ with $\left(\frac{1}{2}, \frac{1}{2}, 0\right)^T$. Thus, we have a contraction. A similar calculation can be performed for each of the subregions, showing that each subregion maps monotonically to an area strictly within another subregion by a contraction map. Figure 3(d) illustrates the various mappings between subregions. After three iterations, each subregion maps by a monotonic contraction to a strict subset of itself. Thus, any fixed point of the three-iteration cycle must be the unique attracting fixed point for that subregion, and the domain of attraction for this point must be the whole subregion. In fact, there are six such fixed points, one for each subregion, three for each of the two cycles. The union of the domains of attraction for these fixed points is the whole triangle; every position $\mathbf{d}$ within the simplex Δ_3 is within the domain of attraction of one of these 3-cycles. Thus, these two cycles are globally stable.

Since the contraction is so strong at every iteration (as shown above, the absolute slope of the map is much less than 1), the convergence to one of these two 3-cycles is very fast. Figure 5(a) shows where each subregion of the "unfolded triangle" will map after the first iteration. The "unfolded triangle" is the interval obtained by traversing the triangle clockwise, starting and ending at $\left(0, \frac{1}{2}, \frac{1}{2}\right)$. Figure 5(b) illustrates that the absolute slope of the second iteration of this map at the fixed points is much less than 1; the cycles are strongly attracting.

The combined classifier that AdaBoost will output is

$$\boldsymbol{\lambda}_{\text{combined}} = \frac{\left(\frac{1}{2}\ln\left(\frac{1+r_1^{cyc}}{1-r_1^{cyc}}\right), \frac{1}{2}\ln\left(\frac{1+r_2^{cyc}}{1-r_2^{cyc}}\right), \frac{1}{2}\ln\left(\frac{1+r_3^{cyc}}{1-r_3^{cyc}}\right)\right)^T}{\text{normalization constant}} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)^T,$$

and since $\min_i(\mathbf{M}\boldsymbol{\lambda}_{\text{combined}})_i = \frac{1}{3}$, we see that AdaBoost always produces a maximum margin solution for this input matrix.

Thus, we have derived our first convergence proof for AdaBoost in a specific separable case. We have shown that at least in some cases, AdaBoost is in fact a margin-maximizing algorithm. We summarize this first main result.

Theorem 1 *For the* 3×3 *matrix* $\boldsymbol{M}$*:*

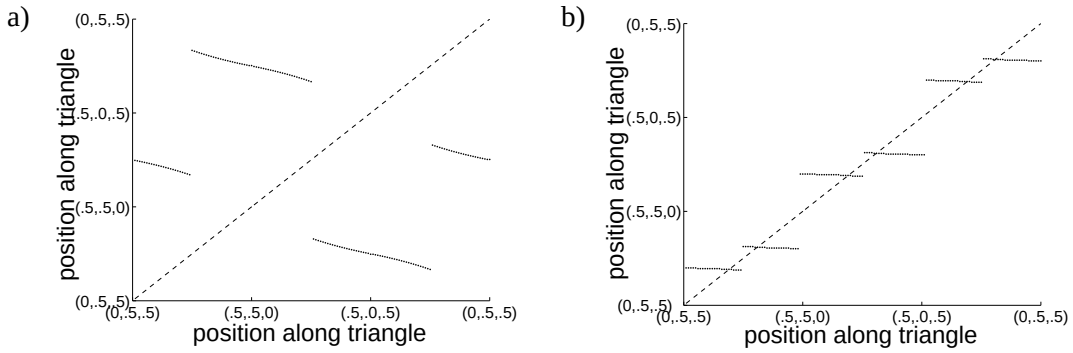


Figure 5: (a) The iterated map on the unfolded triangle. Both axes give coordinates on the edges of the inner triangle in Figure 3(b). The plot shows where $\mathbf{d}_{t+1}$ will be, given $\mathbf{d}_t$. (b) The map from (a) iterated twice, showing where $\mathbf{d}_{t+3}$ will be, given $\mathbf{d}_t$. For this "triple map", there are 6 stable fixed points, 3 for each cycle.

- *The weight vectors* $\mathbf{d}_t$ *converge to one of two possible stable cycles. The coordinates of the cycles are:*

$$\begin{aligned}
\mathbf{d}_1^{cyc} &= \left(\frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4}, \frac{1}{2}\right)^T, \\
\mathbf{d}_2^{cyc} &= \left(\frac{1}{2}, \frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}-1}{4}\right)^T, \\
\mathbf{d}_3^{cyc} &= \left(\frac{\sqrt{5}-1}{4}, \frac{1}{2}, \frac{3-\sqrt{5}}{4}\right)^T,
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{d}_1^{cyc'} &= \left(\frac{3-\sqrt{5}}{4}, \frac{1}{2}, \frac{\sqrt{5}-1}{4}\right)^T, \\
\mathbf{d}_2^{cyc'} &= \left(\frac{1}{2}, \frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4}\right)^T, \\
\mathbf{d}_3^{cyc'} &= \left(\frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4}, \frac{1}{2}\right)^T.
\end{aligned}$$

- *AdaBoost produces a maximum margin solution for this matrix* $\boldsymbol{M}$*.*

5. Generalization to m Classifiers, Each with One Misclassified Example

This simple 3 classifier case can be generalized to m classifiers, each having one misclassified training example; we will find solutions of a similar nature to the ones we found for the 3×3 case,

where there is a rotation of the coordinates at every iteration and a contraction. Here,

$$\mathbf{M} = \begin{pmatrix} -1 & 1 & 1 & \cdots & 1 \\ 1 & -1 & 1 & \cdots & 1 \\ 1 & 1 & -1 & & \vdots \\ \vdots & & & \ddots & 1 \\ 1 & \cdots & \cdots & 1 & -1 \end{pmatrix}.$$

Theorem 2 *For the $m \times m$ matrix above:*

- *The dynamical system for AdaBoost's weight vectors contains at least $(m-1)!$ stable periodic cycles of length m.*
- *AdaBoost converges to a maximum margin solution when the weight vectors converge to one of these cycles.*

The proof of Theorem 2 can be found in Appendix A.

6. Identically Classified Examples and Manifolds of Cycles

In this section, we show how manifolds of cycles appear automatically from cyclic dynamics when there are sets of identically classified training examples. We show that the manifolds of cycles that arise from a variation of the 3×3 case are stable. One should think of a "manifold of cycles" as a continuum of cycles; starting from a position on any cycle, if we move along the directions defined by the manifold, we will find starting positions for infinitely many other cycles. These manifolds are interesting from a theoretical viewpoint. In addition, their existence and stability will be an essential part of the proof of Theorem 7.

A set of training examples $\overline{I}$ is *identically classified* if each pair of training examples i and i' contained in $\overline{I}$ satisfy $y_i h_j(\mathbf{x}_i) = y_{i'} h_j(\mathbf{x}_{i'}) \; \forall j$. That is, the rows i and i' of matrix $\mathbf{M}$ are identical; training examples i and i' are misclassified by the same set of weak classifiers. When AdaBoost cycles, it treats each set of identically classified training examples as one training example, in a specific sense we will soon describe.

For convenience of notation, we will remove the '*cyc*' notation so that $\mathbf{d}_1$ is a position within the cycle (or equivalently, we could make the assumption that AdaBoost starts on a cycle). Say there exists a cycle such that $d_{1,i} > 0 \; \forall i \in \overline{I}$, where $\mathbf{d}_1$ is a position within the cycle and $\mathbf{M}$ possesses some identically classified examples $\overline{I}$. ($\overline{I}$ is not required to include all examples identically classified with $i \in \overline{I}$.) We know that for each pair of identically classified examples i and i' in $\overline{I}$, we have $M_{ij_t} = M_{i'j_t} \; \forall t = 1, ..., T$. Let perturbation $\mathbf{a} \in \mathbb{R}^m$ obey

$$\sum_{i \in \overline{I}} a_i = 0, \text{ and also } a_i = 0 \text{ for } i \notin \overline{I}.$$

Now, let $\mathbf{d}_1^a := \mathbf{d}_1 + \mathbf{a}$. We accept only perturbations $\mathbf{a}$ so that the perturbation does not affect the value of any j_t in the cycle. That is, we assume each component of $\mathbf{a}$ is sufficiently small; since the dynamical system defined by AdaBoost is piecewise continuous, it is possible to choose $\mathbf{a}$ small enough so the perturbed trajectory is still close to the original trajectory after T iterations. Also, $\mathbf{d}_1^a$

must still be a valid distribution, so it must obey the constraint $\mathbf{d}_1^a \in \Delta_m$, i.e., $\sum_i a_i = 0$ as we have specified. Choose any elements i and $i' \in \overline{I}$. Now,

$$\begin{aligned}
r_1^a &= (\mathbf{d}_1^{a\,T}\mathbf{M})_{j_1} = (\mathbf{d}_1{}^T\mathbf{M})_{j_1} + (\mathbf{a}^T\mathbf{M})_{j_1} = r_1 + \sum_{\bar{i}\in\overline{I}} a_{\bar{i}} M_{\bar{i}j_1} = r_1 + M_{i'j_1}\sum_{\bar{i}\in\overline{I}} a_{\bar{i}} = r_1 \\
d_{2,i}^a &= \frac{d_{1,i}^a}{1+M_{ij_1}r_1^a} = \frac{d_{1,i}^a}{1+M_{ij_1}r_1} = \frac{d_{1,i}}{1+M_{ij_1}r_1} + \frac{1}{1+M_{i'j_1}r_1}a_i = d_{2,i} + \frac{1}{1+M_{i'j_1}r_1}a_i \\
r_2^a &= (\mathbf{d}_2^{a\,T}\mathbf{M})_{j_2} = (\mathbf{d}_2{}^T\mathbf{M})_{j_2} + \frac{1}{1+M_{i'j_1}r_1}(\mathbf{a}^T\mathbf{M})_{j_2} = r_2 + \frac{1}{1+M_{i'j_1}r_1}\sum_{\bar{i}\in\overline{I}} a_{\bar{i}} M_{\bar{i}j_2} \\
&= r_2 + \frac{M_{i'j_2}}{1+M_{i'j_1}r_1}\sum_{\bar{i}\in\overline{I}} a_{\bar{i}} = r_2 \\
d_{2,i}^a &= \frac{d_{2,i}^a}{1+M_{ij_2}r_2^a} = \frac{d_{2,i}^a}{1+M_{ij_2}r_2} = \frac{d_{2,i}}{1+M_{ij_2}r_2} + \frac{1}{(1+M_{i'j_2}r_2)(1+M_{i'j_1}r_1)}a_i \\
&= d_{3,i} + \frac{1}{(1+M_{i'j_2}r_2)(1+M_{i'j_1}r_1)}a_i \\
&\vdots \\
d_{T+1,i}^a &= d_{T+1,i} + \frac{1}{\prod_{t=1}^T(1+M_{i'j_t}r_t)}a_i = d_{1,i} + a_i = d_{1,i}^a.
\end{aligned}$$

The cycle condition was used in the last line. This calculation shows that if we perturb any cycle in the directions defined by $\overline{I}$, we will find another cycle. An entire manifold of cycles then exists, corresponding to the possible nonzero acceptable perturbations $\mathbf{a}$. Effectively, the perturbation shifts the distribution among examples in $\overline{I}$, with the total weight remaining the same. For example, if a cycle exists containing vector $\mathbf{d}_1$ with $d_{1,1} = .20, d_{1,2} = .10$, and $d_{1,3} = .30$, where $\{1,2,3\} \subset \overline{I}$, then a cycle with $d_{1,1} = .22, d_{1,2} = .09$, and $d_{1,3} = .29$ also exists, assuming none of the j_t's change; in this way, groups of identically distributed examples may be treated as one example, because they must share a single total weight (again, only within the region where none of the j_t's change).

We will now consider a simple case where manifolds of cycles exist, and we will show that these manifolds are stable in the proof of Theorem 3.

The form of the matrix $\mathbf{M}$ is

$$\begin{pmatrix}
-1 & 1 & 1 \\
\vdots & \vdots & \vdots \\
-1 & 1 & 1 \\
1 & -1 & 1 \\
\vdots & \vdots & \vdots \\
1 & -1 & 1 \\
1 & 1 & -1 \\
\vdots & \vdots & \vdots \\
1 & 1 & -1 \\
1 & 1 & 1 \\
\vdots & \vdots & \vdots \\
1 & 1 & 1
\end{pmatrix}.$$

To be more specific, the first q_1 training examples are misclassified only by h_1, the next q_2 examples are misclassified only by h_2, the next q_3 examples are misclassified only by h_3, and the last q_4

examples are always correctly classified (their weights converge to zero). Thus we consider the components of $\mathbf{d}$ as belonging to one of four pieces; as long as

$$\left(\sum_{i=1}^{q_1} d_i, \sum_{i=q_1+1}^{q_1+q_2} d_i, \sum_{i=q_1+q_2+1}^{q_1+q_2+q_3} d_i\right)^T = \mathbf{d}_1^{cyc}, \mathbf{d}_2^{cyc}, \mathbf{d}_3^{cyc}, \mathbf{d}_1^{cyc'}, \mathbf{d}_2^{cyc'}, \text{ or } \mathbf{d}_3^{cyc'} \text{ from Section 4,}$$

then $\mathbf{d}$ lies on a 3-cycle as we have just shown.

Theorem 3 *For the matrix $\boldsymbol{M}$ defined above, manifolds of cycles exist (there is a continuum of cycles). These manifolds are stable.*

The proof of Theorem 3 can be found in Appendix B.

7. Cycles and Support Vectors

Our goal is to understand general properties of AdaBoost in cases where cycling occurs, to broaden our understanding of the phenomenon we have observed in Sections 4, 5, and 6. Specifically, we show that if cyclic dynamics occur, the training examples with the smallest margin are the training examples whose $d_{t,i}$ values stay non-zero (the "support vectors"). In the process, we provide a formula that allows us to directly calculate AdaBoost's asymptotic margin from the edges at each iteration of the cycle. Finally, we give sufficient conditions for AdaBoost to produce a maximum margin solution when cycling occurs.

As demonstrated in Figure 6, there are many low-dimensional matrices $\mathbf{M}$ for which AdaBoost empirically produces cyclic behavior. The matrices used to generate the cycle plots in Figure 6 are contained in Figure 7. These matrices were generated randomly and reduced (rows and columns that did not seem to play a role in the asymptotic behavior were eliminated). We observe cyclic behavior in many more cases than are shown in the figure; almost every low-dimensional random matrix that we tried (and even some larger matrices) seems to yield cyclic behavior. Our empirical observations of cyclic behavior in many cases leads us to build an understanding of AdaBoost's general asymptotic behavior in cases where cycles exist, though there is not necessarily a contraction at each iteration so the dynamics may be harder to analyze. (We at least assume the cycles AdaBoost produces are stable, since it is not likely we would observe them otherwise.) These cyclic dynamics may not persist in very large experimental cases, but from our empirical evidence, it seems plausible (even likely) that cyclic behavior might persist in cases in which there are very few support vectors.

When AdaBoost converges to a cycle, it "chooses" a set of rows and a set of columns, that is:

- The j_t's cycle amongst some of the columns of $\mathbf{M}$, but not necessarily all of the columns. In order for AdaBoost to produce a maximum margin solution, it must choose a set of columns such that the maximum margin for $\mathbf{M}$ can be attained using only those columns.

- The values of $d_{t,i}$ (for a fixed value of i) are either always 0 or always strictly positive throughout the cycle. A *support vector* is a training example i such that the $d_{t,i}$'s in the cycle are strictly positive. These support vectors are similar to the support vectors of a support vector machine in that they all attain the minimum margin over training examples (as we will show). These are training examples that AdaBoost concentrates the hardest on. The remaining training examples have zero weight throughout the cycle; these are the examples that are easier

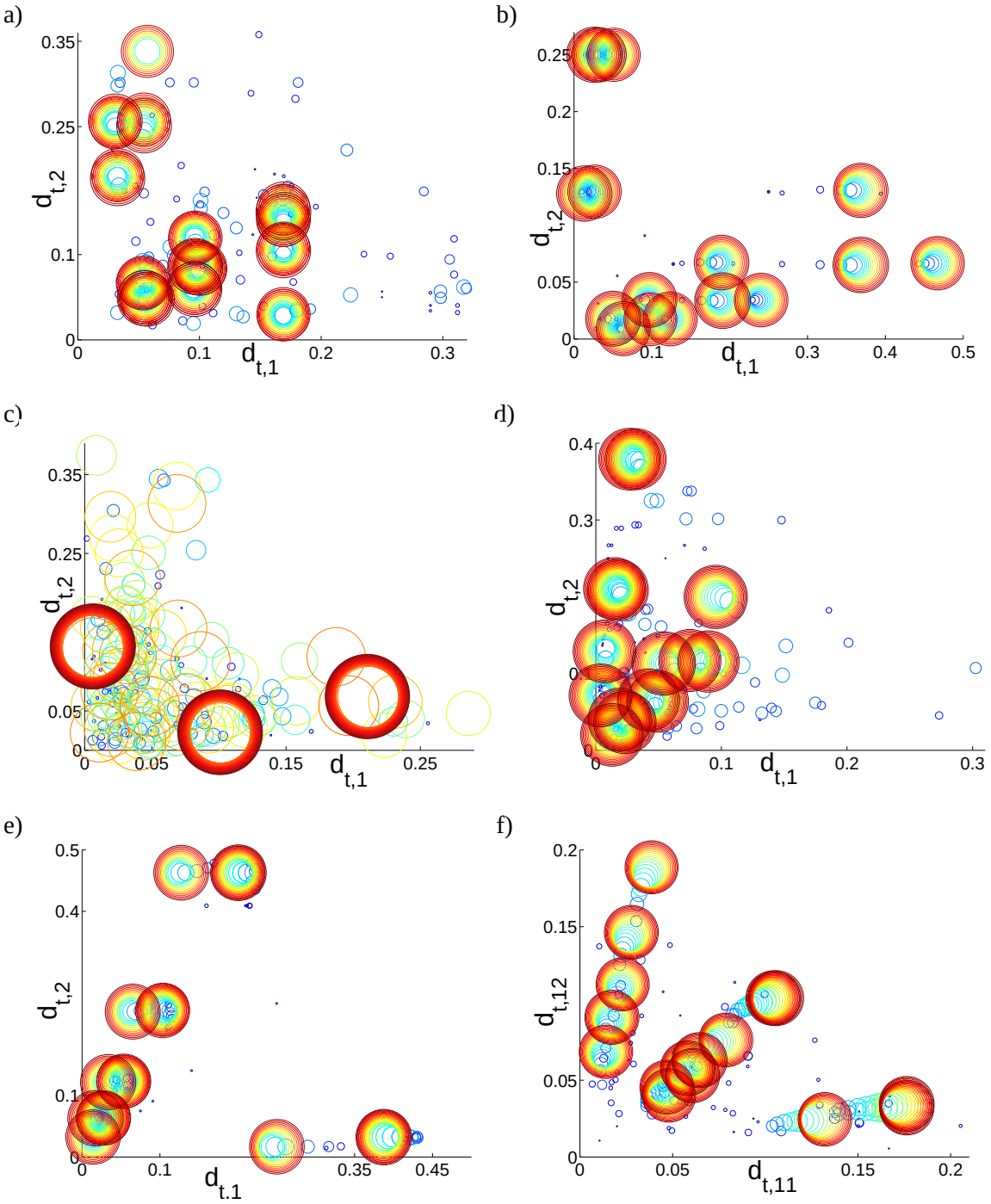


Figure 6: Examples of cycles from randomly generated matrices **M**. An image of **M** for each plot appears in Figure 7. These plots show a projection onto the first two components of AdaBoost's weight vector, e.g., the axes might be $d_{t,1}$ vs. $d_{t,2}$. Smaller circles indicate earlier iterations, and larger circles indicate later iterations. For (a), (d) and (f), 400 iterations were plotted, and for (b) and (e), 300 iterations were plotted. Plot (c) shows 5500 iterations, but only every 20th iteration was plotted. This case took longer to converge, and converged to a simple 3-cycle.

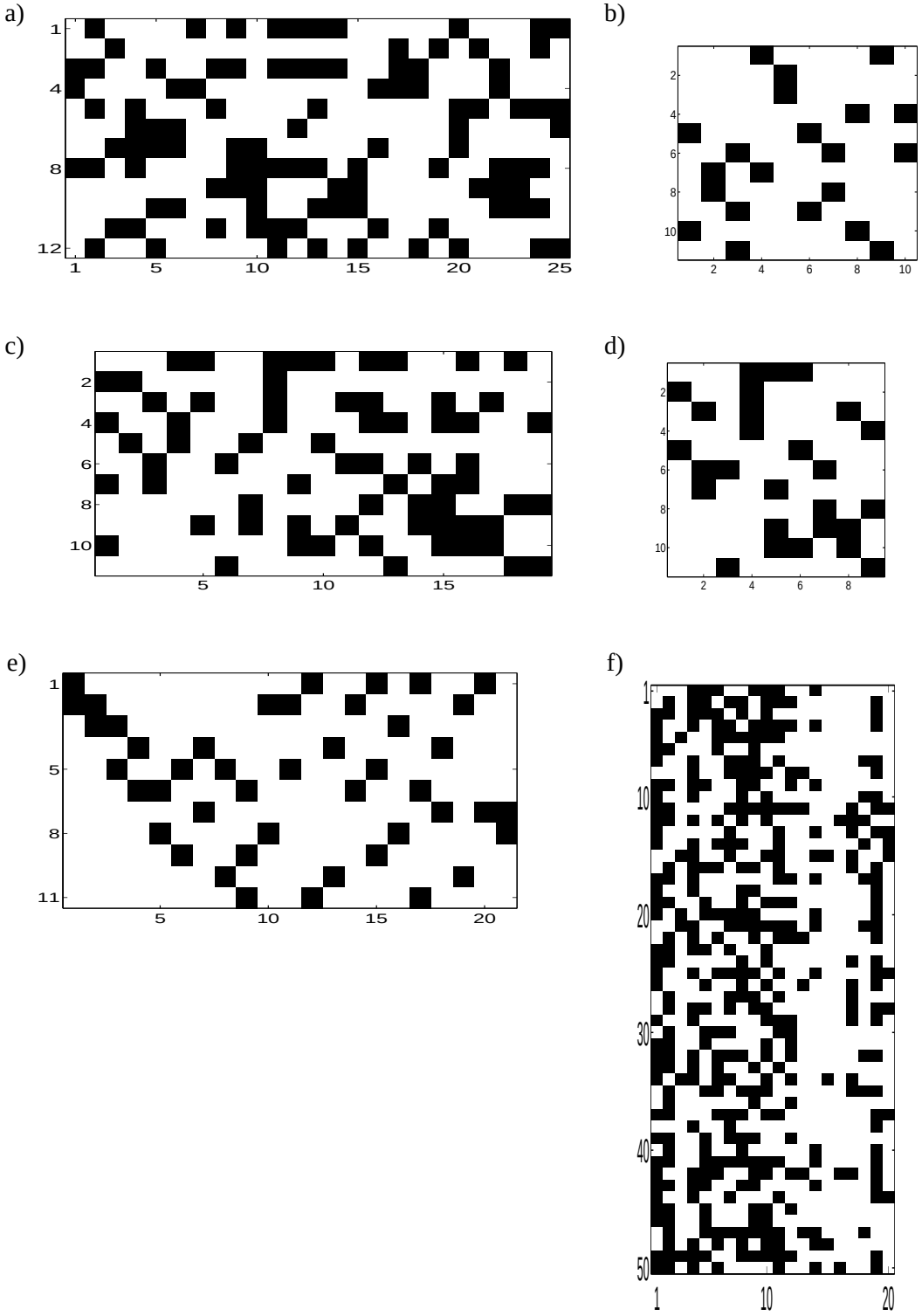


Figure 7: The matrices **M** used to generate the plots in Figure 6. White indicates a value of 1, and black indicates a value of -1. The size of **M** does not seem to have a direct correlation on either the number of iterations per cycle, or the speed of convergence to a cycle.

for the algorithm to classify, since they have margin larger than the support vectors. For support vectors, the cycle condition holds, $\prod_{t=1}^{T}(1+M_{ij_t}r_t^{cyc})=1$. (This holds by Step 3 of the iterated map.) For non-support vectors, $\prod_{t=1}^{T}(1+M_{ij_t}r_t^{cyc})>1$ so the $d_{t,i}$'s converge to 0 (the cycle must be stable).

Theorem 4 *AdaBoost produces the same margin for each support vector and larger margins for other training examples. This margin can be expressed in terms of the cycle parameters* $r_1^{cyc},...,r_T^{cyc}$.

Proof Assume AdaBoost is cycling. Assume $\mathbf{d}_1$ is within the cycle for ease of notation. The cycle produces a normalized output $\boldsymbol{\lambda}^{cyc} := \lim_{t\to\infty}\boldsymbol{\lambda}_t/||\boldsymbol{\lambda}_t||_1$ for AdaBoost. (This limit always converges when AdaBoost converges to a cycle.) Denote

$$z_{cyc} := \sum_{t=1}^{T}\alpha_t = \sum_{t=1}^{T}\frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right).$$

Let i be a support vector. Then,

$$\begin{aligned}
(\mathbf{M}\boldsymbol{\lambda}^{cyc})_i &= \frac{1}{z_{cyc}}\sum_{t=1}^{T}M_{ij_t}\alpha_t = \frac{1}{z_{cyc}}\sum_{t=1}^{T}M_{ij_t}\frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right)\\
&= \frac{1}{2z_{cyc}}\sum_{t=1}^{T}\ln\left(\frac{1+M_{ij_t}r_t}{1-M_{ij_t}r_t}\right) = \frac{1}{2z_{cyc}}\ln\left[\prod_{t=1}^{T}\frac{1+M_{ij_t}r_t}{1-M_{ij_t}r_t}\right]\\
&= \frac{1}{2z_{cyc}}\ln\left[\left(\prod_{t=1}^{T}\frac{1+M_{ij_t}r_t}{1-M_{ij_t}r_t}\right)\left(\prod_{t=1}^{T}\frac{1}{(1+M_{ij_t}r_t)}\right)^2\right]\\
&= \frac{1}{2z_{cyc}}\ln\left[\prod_{t=1}^{T}\frac{1}{(1-M_{ij_t}r_t)(1+M_{ij_t}r_t)}\right]\\
&= \frac{1}{2z_{cyc}}\ln\left[\prod_{t=1}^{T}\frac{1}{1-r_t^2}\right] = -\frac{1}{2}\frac{\ln\prod_{t=1}^{T}(1-r_t^2)}{\sum_{\tilde{t}=1}^{T}\frac{1}{2}\ln\left(\frac{1+r_{\tilde{t}}}{1-r_{\tilde{t}}}\right)}\\
&= -\frac{\ln\prod_{t=1}^{T}(1-r_t^2)}{\ln\prod_{\tilde{t}=1}^{T}\left(\frac{1+r_{\tilde{t}}}{1-r_{\tilde{t}}}\right)}. \qquad (5)
\end{aligned}$$

The first line uses the definition of α_t from the AdaBoost algorithm, the second line uses the fact that $\mathbf{M}$ is binary, the third line uses the fact that i is a support vector, i.e., $\prod_{t=1}^{T}(1+M_{ij_t}r_t)=1$. Since the value in (5) is independent of i, this is the value of the margin that AdaBoost assigns to every support vector i. We denote the value in (5) as μ_{cycle}, which is only a function of the cycle parameters, i.e., the edge values.

Now we show that every non-support vector achieves a larger margin than μ_{cycle}. For a non-support vector i, we have $\prod_{t=1}^{T}(1+M_{ij_t}r_t)>1$, that is, the cycle is stable. Thus, $0>\ln\left[\frac{1}{\prod_{t=1}^{T}(1+M_{ij_t}r_t)}\right]^2$.

Now,

$$\begin{aligned}(\mathbf{M}\boldsymbol{\lambda}^{cyc})_i &= \frac{1}{z_{cyc}}\sum_{t=1}^{T} M_{ij_t}\alpha_t = \frac{1}{2z_{cyc}}\ln\left[\frac{\prod_t(1+M_{ij_t}r_t)}{\prod_{\tilde{t}}(1-M_{ij_{\tilde{t}}}r_{\tilde{t}})}\right] \\ &> \frac{1}{2z_{cyc}}\ln\left[\frac{\prod_t(1+M_{ij_t}r_t)}{\prod_{\tilde{t}}(1-M_{ij_{\tilde{t}}}r_{\tilde{t}})}\right] + \frac{1}{2z_{cyc}}\ln\left[\frac{1}{\prod_{\tilde{t}}(1+M_{ij_{\tilde{t}}}r_{\tilde{t}})}\right]^2 \\ &= \frac{-\ln\prod_{t=1}^{T}(1-r_t^2)}{\ln\prod_{\tilde{t}=1}^{T}\left(\frac{1+r_{\tilde{t}}}{1-r_{\tilde{t}}}\right)} = \mu_{cycle}.\end{aligned}$$

Thus, non-support vectors achieve larger margins than support vectors. ■

The previous theorem shows that the asymptotic margin of the support vectors is the same as the asymptotic margin produced by AdaBoost; this asymptotic margin can be directly computed using (5). AdaBoost may not always produce a maximum margin solution, as we will see in Sections 8 and 9; however, there are sufficient conditions such that AdaBoost will automatically produce a maximum margin solution when cycling occurs. Before we state these conditions, we define the matrix $\mathbf{M}_{cyc} \in \{-1,1\}^{m_{cyc}\times n_{cyc}}$, which contains certain rows and columns of $\mathbf{M}$. To construct $\mathbf{M}_{cyc}$ from $\mathbf{M}$, we choose only the rows of $\mathbf{M}$ that correspond to support vectors (eliminating the others, whose weights vanish anyway), and choose only the columns of $\mathbf{M}$ corresponding to weak classifiers that are chosen in the cycle (eliminating the others, which are never chosen after cycling begins anyway). Here, m_{cyc} is the number of support vectors chosen by AdaBoost, and n_{cyc} is the number of weak classifiers in the cycle.

Theorem 5 *Suppose AdaBoost is cycling, and that the following are true:*

1. $$\max_{\hat{\boldsymbol{\lambda}}\in\Delta_{n_{cyc}}} \min_i (\boldsymbol{M}_{cyc}\hat{\boldsymbol{\lambda}})_i = \max_{\tilde{\boldsymbol{\lambda}}\in\Delta_n} \min_i (\boldsymbol{M}\tilde{\boldsymbol{\lambda}})_i = \rho$$

   (AdaBoost cycles among columns of $\boldsymbol{M}$ that can be used to produce a maximum margin solution.)

2. *There exists $\boldsymbol{\lambda}_\rho \in \Delta_{n_{cyc}}$ such that $(\boldsymbol{M}_{cyc}\boldsymbol{\lambda}_\rho)_i = \rho$ for $i = 1,\ldots,m_{cyc}$. (AdaBoost chooses support vectors corresponding to a maximum margin solution for $\boldsymbol{M}_{cyc}$.)*

3. *The matrix $\boldsymbol{M}_{cyc}$ is invertible.*

Then AdaBoost produces a maximum margin solution.

The first two conditions specify that AdaBoost cycles among columns of $\mathbf{M}$ that can be used to produce a maximum margin solution, and chooses support vectors corresponding to this solution. The first condition specifies that the maximum margin, ρ, (corresponding to the matrix $\mathbf{M}$) must be the same as the maximum margin corresponding to $\mathbf{M}_{cyc}$. Since the cycle is stable, all other training examples achieve larger margins; hence ρ is the best possible margin $\mathbf{M}_{cyc}$ can achieve. The second condition specifies that there is at least one analytical solution $\boldsymbol{\lambda}_\rho$ such that all training examples of $\mathbf{M}_{cyc}$ achieve a margin of exactly ρ.

Proof By Theorem 4, AdaBoost will produce the same margin for all of the rows of $\mathbf{M}_{cyc}$, since they are all support vectors. We denote the value of this margin by μ_{cycle}.

Let $\chi_{m_{cyc}} := (1,1,1,\ldots,1)^T$, with m_{cyc} components. From 2, we are guaranteed the existence of λ_ρ such that

$$\mathbf{M}_{cyc}\lambda_\rho = \rho\chi_{m_{cyc}}.$$

We already know

$$\mathbf{M}_{cyc}\lambda^{cyc} = \mu_{cycle}\chi_{m_{cyc}}$$

since all rows are support vectors for our cycle. Since $\mathbf{M}_{cyc}$ is invertible,

$$\lambda^{cyc} = \mu_{cycle}\mathbf{M}_{cyc}^{-1}\chi_{m_{cyc}} \quad \text{and} \quad \lambda_\rho = \rho\mathbf{M}_{cyc}^{-1}\chi_{m_{cyc}},$$

so we have $\lambda^{cyc} = \text{constant} \cdot \lambda_\rho$. Since λ^{cyc} and λ_ρ must both be normalized, the constant must be 1. Thus $\rho = \mu_{cycle}$. ■

It is possible for the conditions of Theorem 5 not to hold, for example, condition 1 does not hold in the examples of Sections 8 and 9; in these cases, a maximum margin solution is not achieved. It can be shown that the first two conditions are necessary but the third one is not. It is not hard to understand the necessity of the first two conditions; if it is not possible to produce a maximum margin solution using the weak classifiers and support vectors AdaBoost has chosen, then it is not possible for AdaBoost to produce a maximum margin solution. The third condition is thus quite important, since it allows us to uniquely identify λ^{cyc}. Condition 3 does hold for the cases studied in Sections 4 and 5.

8. Non-Optimal AdaBoost Does Not Necessarily Converge to a Maximum Margin Solution, Even if Optimal AdaBoost Does

Based on large scale experiments and a gap in theoretical bounds, Rätsch and Warmuth (2002) conjectured that AdaBoost does not necessarily converge to a maximum margin classifier in the non-optimal case, i.e., that AdaBoost is not robust in this sense. In practice, the weak classifiers are generated by CART or another weak learning algorithm, implying that the choice need not always be optimal.

We will consider a 4×5 matrix $\mathbf{M}$ for which AdaBoost fails to converge to a maximum margin solution if the edge at each iteration is required only to exceed ρ (the non-optimal case). That is, we show that "non-optimal AdaBoost" (AdaBoost in the non-optimal case) may not converge to a maximum margin solution, even in cases where "optimal AdaBoost" does.

Theorem 6 *AdaBoost in the non-optimal case may fail to converge to a maximum margin solution, even if optimal AdaBoost does. An example illustrating this is*

$$\boldsymbol{M} = \begin{pmatrix} -1 & 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 \end{pmatrix}.$$

Proof For this matrix, the maximum margin ρ is 1/2. Actually, in the optimal case, AdaBoost will produce this value by cycling among the first four columns of $\mathbf{M}$. Recall that in the non-optimal case:

$$j_t \in \{j : (\mathbf{d}_t^T \mathbf{M})_j \geq \rho\}.$$

Consider the following initial condition for the dynamics:

$$\mathbf{d}_1 = \left(\frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, \frac{1}{2}, \frac{\sqrt{5}-1}{4} \right)^T.$$

Since $(\mathbf{d}_1^T \mathbf{M})_5 = (\sqrt{5}-1)/2 > 1/2 = \rho$, we are justified in choosing $j_1 = 5$, although here it is not the optimal choice. Another iteration yields

$$\mathbf{d}_2 = \left(\frac{1}{4}, \frac{1}{4}, \frac{\sqrt{5}-1}{4}, \frac{3-\sqrt{5}}{4} \right)^T,$$

satisfying $(\mathbf{d}_1^T \mathbf{M})_4 > \rho$ for which we choose $j_2 = 4$. At the following iteration, we choose $j_3 = 3$, and at the fourth iteration we find $\mathbf{d}_4 = \mathbf{d}_1$. This cycle is the same as one of the cycles considered in Section 4 (although there is one extra dimension). There is actually a whole manifold of 3-cycles, since $\tilde{\mathbf{d}}_1^T := (\varepsilon, \frac{3-\sqrt{5}}{4} - \varepsilon, \frac{1}{2}, \frac{\sqrt{5}-1}{4})$ lies on a (non-optimal) cycle for any ε, $0 \leq \varepsilon \leq \frac{3-\sqrt{5}}{4}$. In any case, the value of the margin produced by this cycle is 1/3, not 1/2. ■

We have thus established that AdaBoost is not robust in the sense we described; if the weak learning algorithm is not required to choose the optimal weak classifier at each iteration, but is required only to choose a sufficiently good weak classifier $j_t \in \{j : (\mathbf{d}_t^T \mathbf{M})_j \geq \rho\}$, a maximum margin solution will not necessarily be attained, even if optimal AdaBoost would have produced a maximum margin solution. We are not saying that the only way for AdaBoost to converge to a non-maximum margin solution is to fall into the wrong cycle; it is conceivable that there may be many other, non-cyclic, ways for the algorithm to fail to converge to a maximum margin solution.

Note that for some matrices $\mathbf{M}$, the maximum value of the margin may still be attained in the non-optimal case; an example is the 3×3 matrix we analyzed in Section 4. If one considers the 3×3 matrix in the non-optimal case, the usual 3-cycle may not persist. Oddly, a 4-cycle may emerge instead. If AdaBoost converges to this 4-cycle, it will still converge to the same (maximum) margin of 1/3. See Appendix C for the coordinates of such a 4-cycle. Thus, there is no guarantee as to whether the non-optimal case will produce the same asymptotic margin as the optimal case.

In Figure 8, we illustrate the evolution of margins in the optimal and non-optimal cases for matrix $\mathbf{M}$ of Theorem 6. Here, optimal AdaBoost converges to a margin of $1/2$ via convergence to a 4-cycle, and non-optimal AdaBoost converges to a margin of 1/3 via convergence to a 3-cycle.

9. Optimal AdaBoost Does Not Necessarily Converge to a Maximum Margin Solution

In this section, we produce a low-dimensional example that answers the question of whether AdaBoost always converges to a maximum margin in the optimal case.

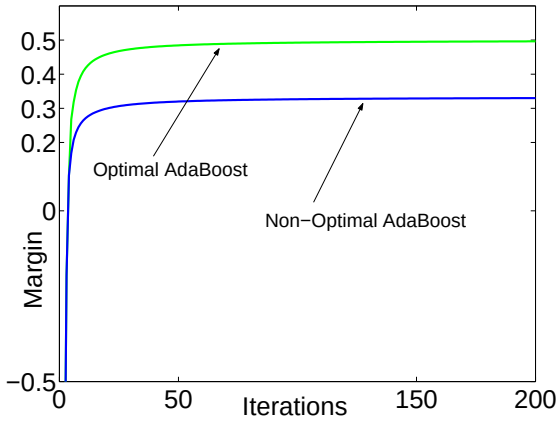


Figure 8: AdaBoost in the optimal case (higher curve) and in the non-optimal case (lower curve). Optimal AdaBoost converges to a margin of $1/2$ via convergence to a 4-cycle, and non-optimal AdaBoost converges to a margin of 1/3 via convergence to a 3-cycle. In both cases we start with $\boldsymbol{\lambda}_1 = \mathbf{0}$.

Theorem 7 *Consider the following matrix whose image appears in Figure 9 (one can see the natural symmetry more easily in the imaged version):*

$$\boldsymbol{M} = \begin{bmatrix} -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}. \tag{6}$$

For this matrix, it is possible for AdaBoost to fail to converge to a maximum margin solution.

Proof The dynamical system corresponding to this matrix contains a manifold of strongly attracting 3-cycles. The cycles we will analyze alternate between weak classifiers 3, 2, and 1. If we consider only weak classifiers 1, 2, and 3, we find that training examples $i = 1$ and 2 are identically classified, i.e., rows 1 and 2 of matrix $\mathbf{M}$ are the same (only considering columns 1, 2, and 3). Similarly, examples 3, 4 and 5 are identically classified, and additionally, examples 6 and 7. Training example 8 is correctly classified by each of these weak classifiers. Because we have constructed $\mathbf{M}$ to have such a strong attraction to a 3-cycle, there are many initial conditions (initial values of $\boldsymbol{\lambda}$) for which AdaBoost will converge to one of these cycles, including the vector $\boldsymbol{\lambda} = \mathbf{0}$. For the first iteration, we chose $j_t = 1$ to achieve the cycle we will analyze below; there are a few different choices for j_t within the first few iterations, since the argmax set sometimes contains more than one element. The dynamics may converge to a different 3-cycle, depending on which values of j_t are chosen within the first few iterations. (Oddly enough, there are initial values of $\boldsymbol{\lambda}$ where AdaBoost converges to

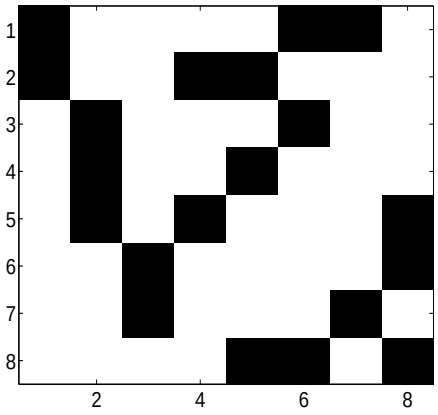


Figure 9: The image of the matrix **M** in (6). White indicates +1, black indicates -1. This matrix has natural symmetry.

a cycle in which a maximum margin solution is produced, although finding such a cycle requires some work.)

To show that a manifold of 3-cycles exists, we present a vector $\mathbf{d}_1$ such that $\mathbf{d}_4 = \mathbf{d}_1$, namely:

$$\mathbf{d}_1 = \left(\frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{\sqrt{5}-1}{8}, \frac{\sqrt{5}-1}{8}, 0 \right)^T. \tag{7}$$

To see this, we iterate the iterated map 4 times.

$$\mathbf{d}_1^T \mathbf{M} = \left(\frac{\sqrt{5}-1}{2}, 0, \frac{3-\sqrt{5}}{2}, \frac{3\sqrt{5}-1}{12}, \frac{3\sqrt{5}-1}{12}, \frac{3\sqrt{5}-1}{12}, \frac{1}{2}, \frac{11-3\sqrt{5}}{12} \right),$$

and here $j_1 = 1$,

$$\begin{aligned} \mathbf{d}_2 &= \left(\frac{1}{4}, \frac{1}{4}, \frac{\sqrt{5}-1}{12}, \frac{\sqrt{5}-1}{12}, \frac{\sqrt{5}-1}{12}, \frac{3-\sqrt{5}}{8}, \frac{3-\sqrt{5}}{8}, 0 \right)^T \\ \mathbf{d}_2^T \mathbf{M} &= \left(0, \frac{3-\sqrt{5}}{2}, \frac{\sqrt{5}-1}{2}, \frac{4-\sqrt{5}}{6}, \frac{4-\sqrt{5}}{6}, \frac{4-\sqrt{5}}{6}, \frac{\sqrt{5}-1}{4}, \frac{5+\sqrt{5}}{12} \right), \end{aligned}$$

and here $j_2 = 3$,

$$\begin{aligned} \mathbf{d}_3 &= \left(\frac{\sqrt{5}-1}{8}, \frac{\sqrt{5}-1}{8}, \frac{3-\sqrt{5}}{12}, \frac{3-\sqrt{5}}{12}, \frac{3-\sqrt{5}}{12}, \frac{1}{4}, \frac{1}{4}, 0 \right)^T \\ \mathbf{d}_3^T \mathbf{M} &= \left(\frac{3-\sqrt{5}}{2}, \frac{\sqrt{5}-1}{2}, 0, \frac{3}{4} - \frac{\sqrt{5}}{12}, \frac{3}{4} - \frac{\sqrt{5}}{12}, \frac{3}{4} - \frac{\sqrt{5}}{12}, \frac{3-\sqrt{5}}{4}, \frac{\sqrt{5}}{6} \right), \end{aligned}$$

and here, $j_3 = 2$, and then $\mathbf{d}_4 = \mathbf{d}_1$.

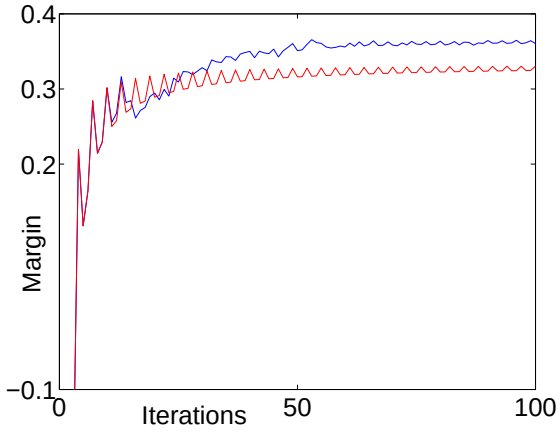


Figure 10: AdaBoost (lower curve) and Approximate Coordinate Ascent Boosting (higher curve), using the 8×8 matrix $\mathbf{M}$ given in Section 9 and initial condition $\boldsymbol{\lambda} = \mathbf{0}$. AdaBoost converges to a margin of 1/3, yet the value of ρ is 3/8. Thus, AdaBoost does not converge to a maximum margin solution for this matrix $\mathbf{M}$.

Hence the 3-cycle exists, and since there are identically classified examples, a manifold of cycles exists; it is automatically stable due to the calculation in the proof of Theorem 3. The margin produced by one of these 3-cycles is always 1/3, yet the maximum margin for this matrix is 3/8. To see that a margin of 3/8 can be obtained, note that $\mathbf{M} \times [2,3,4,1,2,2,1,1]^T \times 1/16 = 3/8$ for all i.■

In Figure 10, we have plotted the evolution of the margin over time for $\mathbf{M}$, for both AdaBoost and Approximate Coordinate Ascent Boosting. Approximate Coordinate Ascent Boosting (Rudin et al., 2004c,b,a; Rudin, 2004) is an algorithm similar to AdaBoost that converges to a maximum margin solution, and runs in polynomial time. AdaBoost rapidly converges to the cycle analyzed above and does not produce a maximum margin solution.

Again, we are not saying that the only way for AdaBoost to converge to a non-maximum margin solution is to fall into the wrong cycle since there may be many other non-cyclic ways for the algorithm to fail to converge to a maximum margin solution. However, many high dimensional cases can be reduced to low dimensional cases simply by restricting our attention to support vectors and weak classifiers that are actually chosen by the algorithm. Thus, a "bad" cycle may not be as uncommon as one would expect, even in a realistic setting.

10. Indications of Chaos

Although we do observe cyclic behavior for many random low-dimensional matrices, we have found an example for which AdaBoost exhibits behavior resembling that of a chaotic dynamical system. In particular, this case exhibits:

- Sensitivity to initial conditions.

- Movement into and out of cyclic behavior.

The matrix **M** we consider for this section is given in Figure 7(a).

Figure 11 shows AdaBoost's edge r_t for t ranging from 0 to 10,000; a number of different initial conditions were considered, which are of the form $\lambda_{1,i} = a$ for all i, for $a = 0, 0.01, 0.02, 0.03, 0.05, 0.06, 0.07, 0.08, 0.09$ and 0.1, (a-j) respectively. In many of these cases, cyclic behavior occurs after some time. In fact, for $a = 0.08$, AdaBoost converges to a 3-cycle. Sometimes, AdaBoost seems to migrate in and out of cyclic behavior, and its behavior is not at all clear. Thus, this example suggests sensitivity to initial conditions. This sensitivity makes sense, since the iterated map is not continuous, it is only *piecewise* continuous. That is, if the argmax set contains two different elements, say j_1 and j_2, the arbitrary choice between them may cause the dynamics to change spectacularly. If we are near such a boundary of a j_t region, a small perturbation may change the choice of j_t chosen and the trajectory may change dramatically. (Note that the Li and Yorke "Period-3-Implies-Chaos" result does not apply to the dynamical system defined by AdaBoost since the iterated map is not continuous, as illustrated in Figure 5 for the 3×3 case.)

Within Figure 11, we can see AdaBoost moving into and out of cyclic behavior, for example, in Figure 11(j). In order to closely examine the switch between the large region of cycling within approximately iterations 8500-9381 and the following chaotic region, we focus our attention on the iterations just before the switch into chaotic behavior. This switch does seem to occur due to a change in region. In other words, as AdaBoost cycles for many iterations (in a cycle of length 14), the weight vectors (viewed every 14th iteration, as in Figure 12) migrate towards the edge of a region and eventually cross over this edge. Where previously, at every 14^{th} iteration AdaBoost would choose $j_t = 19$, it instead chooses $j_t = 3$. Figure 12 shows the values of $(\mathbf{d}_t^T\mathbf{M})_3$ and $(\mathbf{d}_t^T\mathbf{M})_{19}$ at every 14^{th} iterate preceding the switch into chaotic behavior at iteration 9381. Figure 13, which shows the evolution of two components of the weight vector, also illustrates the switches into and out of chaotic behavior.

Eventually, the dynamics drift back towards the same cycle and actually seem to converge to it, as shown in Figure 14(a). Here, the weight vectors do not cross regions, since the values of the largest two components of $(\mathbf{d}^T\mathbf{M})$ do not cross, as shown in Figure 14(b).

Thus, there are many open questions regarding AdaBoost's dynamics that could help us understand its asymptotic behavior; for example, is AdaBoost chaotic in some cases or does it perhaps always produce cyclic behavior asymptotically?

11. Conclusions

We have used the nonlinear iterated map defined by AdaBoost to understand its update rule in low-dimensional cases and uncover remarkable cyclic dynamics. We describe many aspects of AdaBoost's dynamical traits including the fact that AdaBoost does not necessarily converge to a maximum margin solution. We have also proved the conjecture that AdaBoost is not robust to the choice of weak classifier. The key to answering these questions was our analysis of cases in which AdaBoost's asymptotic behavior could be completely determined. Thus an understanding of simple cases has yielded answers to important open questions concerning AdaBoost's large-scale asymptotic behavior.

We leave open many interesting questions. To what extent is AdaBoost chaotic? For what cases does AdaBoost produce maximum margin solutions? Does AdaBoost always exhibit cyclic behavior in the limit? If AdaBoost can behave chaotically without converging to a cycle, how large does the

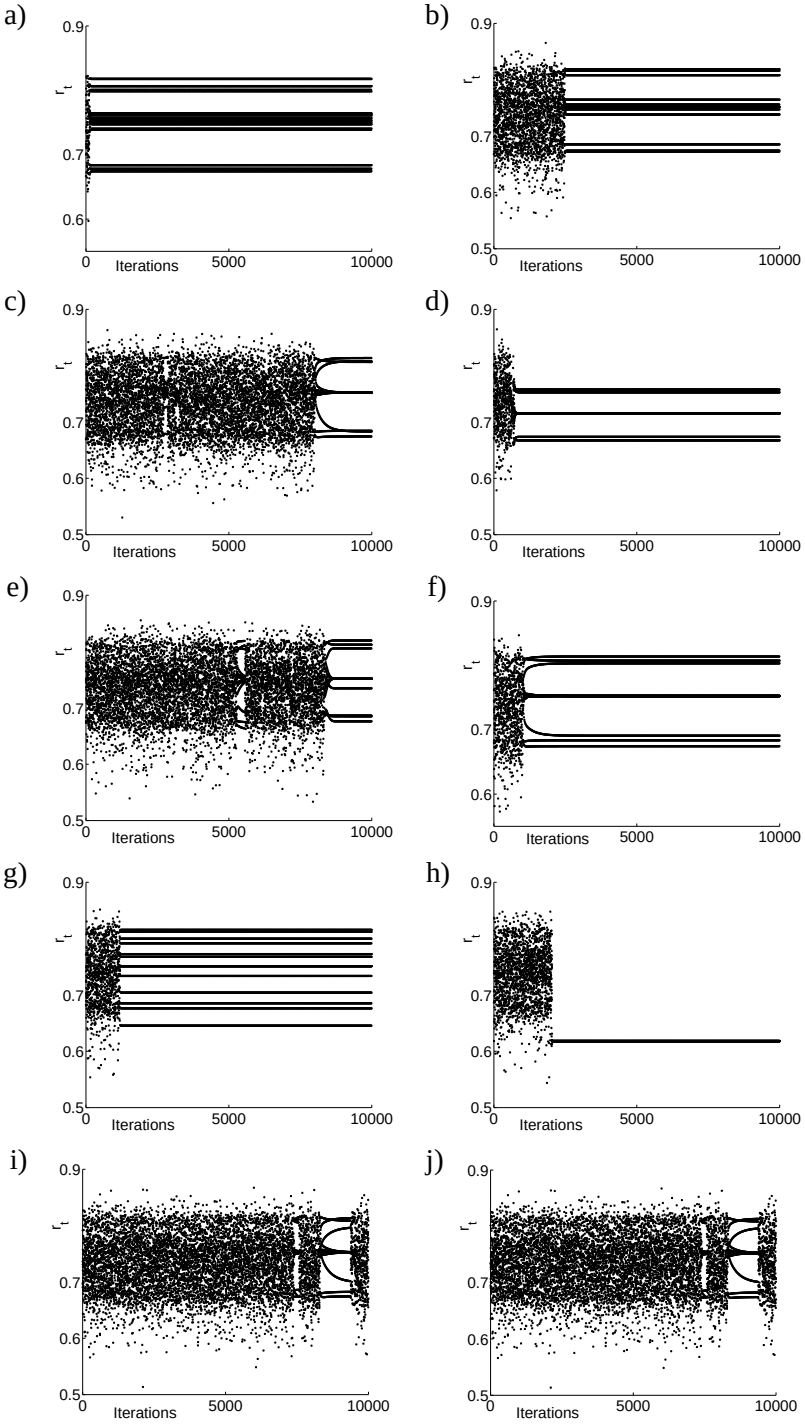


Figure 11: AdaBoost is sensitive to initial conditions. Value of the edge r_t at each iteration t, for many different runs of AdaBoost. For all plots we used the matrix $\mathbf{M}$ shown in Figure 7(a), but with slightly different initial conditions. Some of these plots look somewhat chaotic except for a few regions where AdaBoost seems to be converging to a cycle before becoming chaotic again. In (h), AdaBoost converges to a simple 3-cycle after a significant number of iterations.

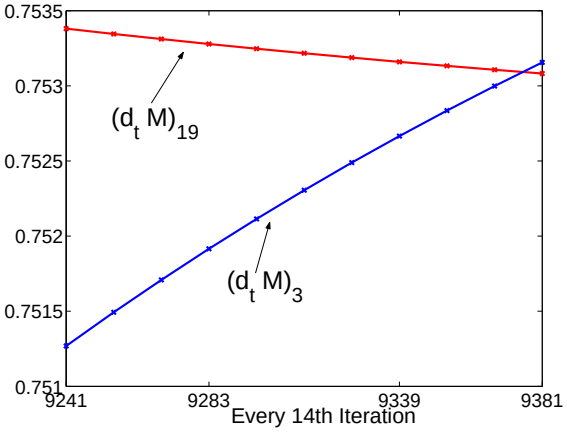


Figure 12: The values of $(\mathbf{d}_t^T\mathbf{M})_3$ and $(\mathbf{d}_t^T\mathbf{M})_{19}$ at every 14^{th} iterate preceding the switch into chaotic behavior of Figure 11(j) where $a = 0.1$. AdaBoost switches from a 14-cycle into chaotic behavior after iteration 9381 when it switches regions, from the region where $j_t = 19$ into the region where $j_t = 3$.

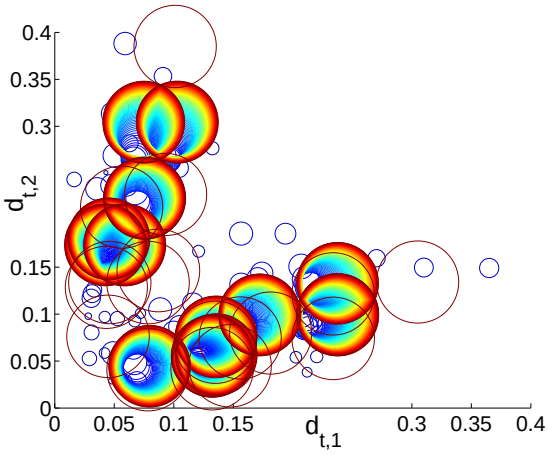


Figure 13: Scatter plot of $d_{t,1}$ vs. $d_{t,2}$ for the iterations surrounding the slow convergence to the cycle in Figure 11(j), where the initial condition is $\lambda_{1,j} = 0.1$ for all j. By examining the circles (especially the smallest and largest ones) one can see the switch into the cyclic behavior, the slow migration towards a cycle, and the fast switch back into chaotic behavior. Again, smaller circles indicate earlier iterations and larger circles indicate later iterations.

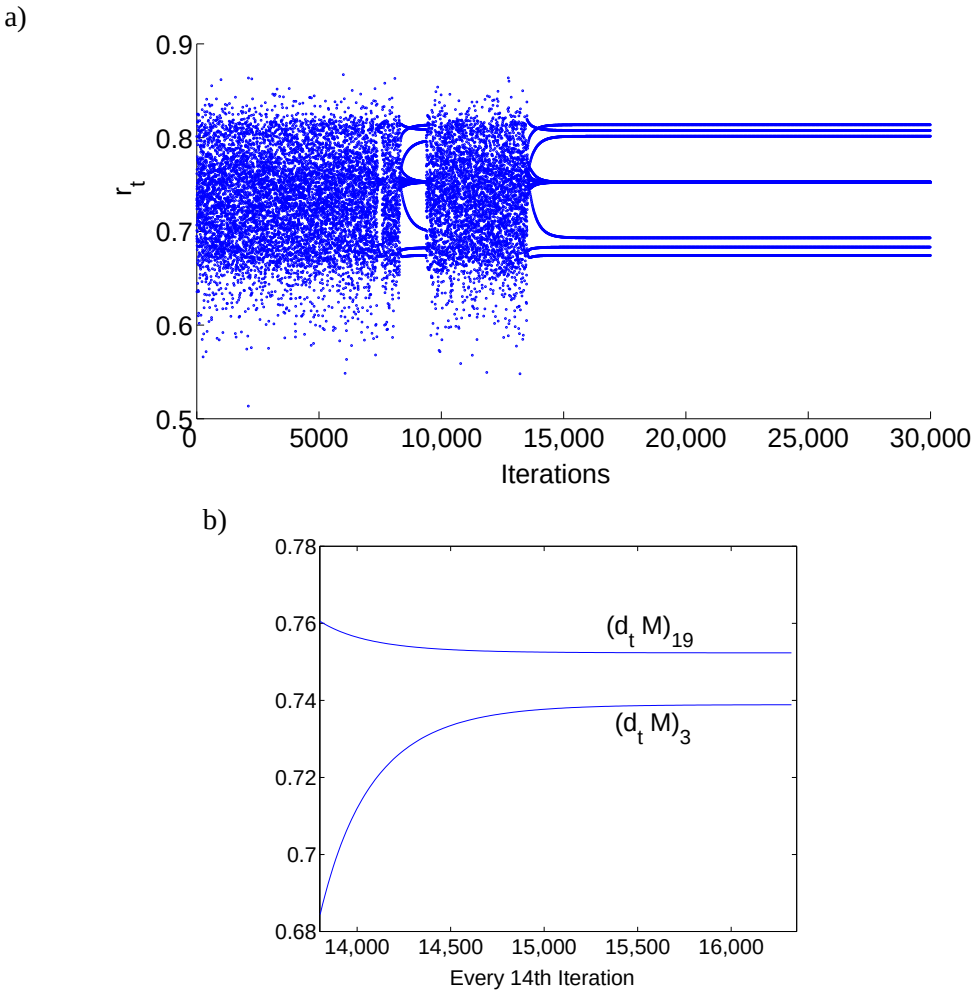


Figure 14: (a) This is the same plot as in Figure 11(j), extended to 30,000 iterations. After more than 13,000 iterations, AdaBoost finally seems to settle on the 14-cycle. (b) The same plot as in Figure 12, but for a different set of iterations. In Figure 12 the edges corresponding to $j = 19$ and $j = 3$ cross, whereas here, the edges are well separated. Thus, AdaBoost is able to maintain the cycle.

matrix **M** need to be in order to produce such behavior? And finally, how does AdaBoost achieve its strong generalization performance if it does not simply maximize the margin? We hope the analytical tools provided in this work, namely the reduction of AdaBoost to a dynamical system and the analysis of its asymptotic behavior in special cases, will help yield answers to these interesting questions.

Acknowledgments

This material is based upon work partially supported by the National Science Foundation under grant numbers CCR-0325463, IIS-0325500 and DMS-9810783. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work is also partially supported by AFOSR award F49620-01-1-0099.

Appendix A. Proof of Theorem 2

Let us first assume a cycle with rotating coordinates exists for this case, and then we will prove its existence and stability. Denote the first position in our cycle as follows (we drop the (cyc) notation here):

$$\mathbf{d}_1 = (\beta_1, \cdots, \beta_i, \cdots, \beta_{m-1}, \beta_m)^T,$$

where $0 < \beta_1 < \beta_2 < \cdots < \beta_m$. Note that $\beta_m = \frac{1}{2}$, again because $(\mathbf{d}_2^T \mathbf{M})_1 = 0$ and $\sum_i d_{2,i} = 1$. Now,

$$\mathbf{d}_1^T \mathbf{M} = ((1-2\beta_1), \cdots, (1-2\beta_{m-1}), 0)^T,$$

so $j_1 = 1$ and $r_1 = 1 - 2\beta_1$. Then, using the iterated map,

$$\mathbf{d}_2 = \left(\frac{1}{2}, \frac{\beta_2}{2(1-\beta_1)}, \cdots, \frac{\beta_i}{2(1-\beta_1)}, \cdots, \frac{\beta_{m-1}}{2(1-\beta_1)}, \frac{\beta_m}{2(1-\beta_1)} \right)^T.$$

Assuming that the coordinates cycle,

$$\beta_{i-1} = \frac{\beta_i}{2(1-\beta_1)} \quad \text{for} \quad i = 2, ..., m, \tag{8}$$

in order for the current iterate to agree with the next iterate. This recursive relation gives

$$\beta_{i-(i-1)} = \frac{\beta_i}{[2(1-\beta_1)]^{i-1}},$$

so that

$$\beta_1 = \frac{\beta_m}{[2(1-\beta_1)]^{m-1}} \tag{9}$$

and

$$\beta_i = \beta_1 [2(1-\beta_1)]^{i-1}, \quad \text{for } i = 1, ..., m. \tag{10}$$

Thus, substituting (9) into (10), recalling that $\beta_m = 1/2$,

$$\beta_i = \frac{\beta_m}{[2(1-\beta_1)]^{m-1}} [2(1-\beta_1)]^{i-1} = \frac{1}{2} [2(1-\beta_1)]^{i-m}. \tag{11}$$

It remains to show that there is a viable solution for β_1 to prove the existence of a cycle. (We require a solution to obey $\beta_1 \leq 1/m$ so that it is possible for $\sum_i d_{1,i} = 1$.) The condition $\sum_i d_{1,i} = 1$ can be rewritten as

$$1 = \frac{1}{2}\sum_{i=1}^{m}[2(1-\beta_1)]^{i-m}.$$

Substituting $\varsigma = 2(1-\beta_1)$ and multiplying both sides by $2\varsigma^{m-1}$, we have a geometric series:

$$2\varsigma^{m-1} = \sum_{i=1}^{m}\varsigma^{i-m+m-1} = \sum_{i=1}^{m}\varsigma^{i-1} = \frac{1-\varsigma^m}{1-\varsigma},$$

that is,

$$\varsigma^m - 2\varsigma^{m-1} + 1 = 0. \tag{12}$$

Substituting back for ς,

$$2^m(1-\beta_1)^{m-1}[(1-\beta_1)-1]+1 = 0,$$

or more simply,

$$1 - \beta_1 2^m (1-\beta_1)^{m-1} = 0.$$

To show a solution exists for $m \geq 4$ (we have already handled the $m = 3$ case), we will apply the Intermediate Value Theorem to the function

$$\varphi(\bar{\beta_1}, m) := 1 - \bar{\beta_1} 2^m (1-\bar{\beta_1})^{m-1}.$$

We know $\varphi(0,m) = 1 > 0$. Consider

$$\varphi\left(\frac{1}{10}, m\right) = 1 - 2^m \frac{1}{10}\left(\frac{9}{10}\right)^{m-1} = 1 - \left(\frac{9}{5}\right)^m \frac{1}{9}.$$

Plugging in $m = 4$, we find that $\varphi(1/10, 4) = -104/625 < 0$. On the other hand, extending the definition of φ to non-integer values of m, we have

$$\frac{\partial\varphi(1/10,m)}{\partial m} = -\frac{1}{9}\left[\ln\left(\frac{9}{5}\right)\right]\left(\frac{9}{5}\right)^m < 0 \quad \text{for all } m \geq 4.$$

Hence, $\varphi(1/10, m) \leq -104/625 < 0$ for all $m \geq 4$. By the Intermediate Value Theorem, there is a root β_1 of $\varphi(\cdot, m)$ for any $m \geq 4$ with $0 \leq \beta_1 \leq 1/10$. Since

$$\frac{\partial\varphi(\bar{\beta_1}, m)}{\partial\bar{\beta_1}} = 2^m(1-\bar{\beta_1})^{m-2}(m\bar{\beta_1} - 1),$$

we have $\frac{\partial\varphi}{\partial\beta_1}(\bar{\beta_1}, m) = 0$ only when $\bar{\beta_1} = 1/m$, and that $\varphi(\cdot, m)$ decreases for $0 \leq \bar{\beta_1} < 1/m$ and increases for $\bar{\beta_1} > 1/m$ (where $\bar{\beta_1} < 1$). If $m \leq 10$, since $\varphi(\cdot, m)$ decreases for $0 \leq \bar{\beta_1} < 1/m$, the Intermediate Value Theorem provides the unique root $0 \leq \beta_1 \leq 1/10 \leq 1/m$. If $m > 10$, $\varphi(\cdot, m)$ decreases for $0 \leq \beta_1 < 1/m$ and increases to the value $\varphi(1/10, m)$, which is negative. Thus, there is a unique root $0 \leq \beta_1 \leq 1/m$. Hence, the root exists and is unique for $m \geq 4$. Now we have shown the existence and uniqueness of our cycle, namely the cycle starting from

$$\mathbf{d}_1 = \left(\beta_1, 2\beta_1(1-\beta_1), 4\beta_1^2(1-\beta_1)^2, \ldots, 1/2\right)^T.$$

Of course, this is not the only periodic orbit. Any permutation of the components in $\mathbf{d}_1$ will lie on a periodic cycle. If (without loss of generality) we fix the first iteration of each cycle to start with the same first component $d_{1,i} = \beta_1$, then the number of permutations of the other components (and thus the number of periodic cycles we have defined by relabelling the coordinates) is $(m-1)!$.

We now show that these $(m-1)!$ cycles are stable. It is sufficient to show that just one cycle is stable, since the others are obtained by simply relabelling the order of the coordinates (without loss of generality say $j_t = t$ for $t = 1, ..., m$). We add a perturbation $\varepsilon\mathbf{a}$ to $\mathbf{d}_1$, small enough so that none of the j_t's chosen within the cycle are affected. (Note that choosing such a perturbation is possible, since the map is piecewise continuous, and $\beta_1 < ... < \beta_m$ without equality between the β_i's. This will ensure that no $\mathbf{d}_t$ lies on the boundary of a region, so that the $\text{argmax}_j(\mathbf{d}_t^T\mathbf{M})_j$ set contains exactly one element.) Also, we require $\sum_{i=1}^m a_i = 0$ so the perturbed starting point still lies on the simplex Δ_m. Assume $\|\mathbf{a}\|_1$ is $O(1)$, and that ε is small. Our perturbed starting point is

$$\mathbf{d}_1^a := \mathbf{d}_1 + \varepsilon\mathbf{a}.$$

Now, since $r_1 = 1 - 2\beta_1$,

$$r_1^a \quad = \quad (\mathbf{d}_1^{aT}\mathbf{M})_1 = (\mathbf{d}_1{}^T\mathbf{M})_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1 = 1 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1.$$

Again we use the iterated map. Recall that $\beta_m = \frac{1}{2}$, and $d_{2,1}^a = \frac{1}{2}$. For all other i,

$$d_{2,i}^a = \frac{\beta_i + \varepsilon a_i}{1 + r_1^a} = \frac{\beta_i + \varepsilon a_i}{2 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1}.$$

To see whether the perturbation has shrunk due to the dynamics, we compute $\varepsilon\tilde{\mathbf{a}} := \mathbf{d}_2^a - \mathbf{d}_2$. If $\|\tilde{\mathbf{a}}\|_1 \leq C\|\mathbf{a}\|_1$ where C is a constant less than 1, the map is a contraction. Recall that

$$\mathbf{d}_2 = \left(\frac{1}{2}, \beta_1, \cdots, \beta_{m-1}\right)^T.$$

Thus,

$$\varepsilon\tilde{a}_1 = d_{2,1}^a - d_{2,1} = 0,$$

and for other i,

$$\varepsilon\tilde{a}_i = d_{2,i}^a - d_{2,i} = \frac{\beta_i + \varepsilon a_i}{2 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1} - \beta_{i-1}.$$

We are done with the $i = 1$ term. For all other terms, we will do an approximation to first order in ε. Using a first order Taylor expansion $\frac{1}{1+x} \approx 1 - x$, we obtain

$$\frac{1}{1 + r_1^a} = \frac{1}{2 - 2\beta_1 + \varepsilon(\mathbf{a}^T\mathbf{M})_1} = \frac{1}{2(1-\beta_1)\left(1 + \frac{\varepsilon(\mathbf{a}^T\mathbf{M})_1}{2(1-\beta_1)}\right)} \approx \frac{1 - \frac{\varepsilon(\mathbf{a}^T\mathbf{M})_1}{2(1-\beta_1)}}{2(1-\beta_1)}.$$

Our expansion yields:

$$\varepsilon\tilde{a}_i = d_{2,i}^a - d_{2,i} \quad \approx \quad \frac{(\beta_i + \varepsilon a_i)\left(1 - \frac{\varepsilon(\mathbf{a}^T\mathbf{M})_1}{2(1-\beta_1)}\right)}{2(1-\beta_1)} - \beta_{i-1}.$$

Grouping terms in orders of ε, we can use (8) to show the first term vanishes, and we find:

$$\begin{aligned}\varepsilon\tilde{a}_i &\approx 0+\varepsilon\left(\frac{a_i}{2(1-\beta_1)}-\frac{\beta_i(\mathbf{a}^T\mathbf{M})_1}{4(1-\beta_1)^2}\right)+O(\varepsilon^2), \quad \text{so that}\\ \tilde{a}_i &\approx \frac{a_i}{2(1-\beta_1)}-\frac{\beta_i(\mathbf{a}^T\mathbf{M})_1}{4(1-\beta_1)^2}+O(\varepsilon).\end{aligned}$$

We will show that the perturbation shrinks at every iteration. Since ε is small, we do not care about the $O(\varepsilon)$ contribution to $\tilde{a}_i$. Recall that $\sum_i \beta_i = 1$, so that:

$$\begin{aligned}\|\tilde{\mathbf{a}}\|_1 &\leq \frac{1}{2(1-\beta_1)}\sum_{i=1}^m |a_i| + \frac{\left|(\mathbf{a}^T\mathbf{M})_1\right|}{4(1-\beta_1)^2}\sum_{i=1}^m \beta_i + O(\varepsilon)\\ &= \frac{1}{2(1-\beta_1)}\|\mathbf{a}\|_1 + \frac{\left|(\mathbf{a}^T\mathbf{M})_1\right|}{4(1-\beta_1)^2} + O(\varepsilon)\\ &\leq \frac{1}{2(1-\beta_1)}\|\mathbf{a}\|_1 + \frac{1}{4(1-\beta_1)^2}\|\mathbf{a}\|_1 + O(\varepsilon)\\ &= \frac{3-2\beta_1}{4(1-\beta_1)^2}\|\mathbf{a}\|_1 + O(\varepsilon).\end{aligned}$$

For the third line, we used the fact that the entries of $\mathbf{M}$ are always within $\{-1,+1\}$. In order to have

$$\frac{3-2\beta_1}{4(1-\beta_1)^2} < 1,$$

we would need

$$3-2\beta_1 < 4(1-\beta_1)^2 = 4-8\beta_1+4\beta_1^2,$$

i.e.,

$$0 < 1-6\beta_1+4\beta_1^2,$$

or more simply,

$$\beta_1 < (3-\sqrt{5})/4.$$

This condition does hold, since

$$0 \leq \beta_1 \leq 1/10 < (3-\sqrt{5})/4.$$

Thus we have shown a contraction of the perturbation at the first iteration. An identical calculation (one must simply reorder the components in the vectors) will yield a contraction at every iteration, so our cycle is stable. We have thus proven the existence and stability of $(m-1)!$ cycles for the case with m weak classifiers, each with 1 misclassified example.

Appendix B. Proof of Theorem 3

The existence of manifolds of cycles follows from the fact that $\mathbf{M}$ has the same 3-cycles as the 3×3 case, except that the weight is distributed among identically classified examples. (Recall that the weights of the last q_4 examples vanish, since these examples are always correctly classified.) In order to move along the manifold, just shift the weights among identically classified examples; this new weight vector will lie directly on another 3-cycle.

In order to show the manifold is stable, we will show that any vector $\mathbf{d}^a$ that lies sufficiently near the manifold will be attracted towards it. More specifically, we will:

- choose an arbitrary vector $\mathbf{d}_1^a$ sufficiently close to the manifold of stable cycles.
- carefully choose a corresponding vector $\mathbf{d}_1$ on the manifold.
- show there is a contraction, sending the successive $\mathbf{d}_t^a$ vectors closer to the $\mathbf{d}_t$ vectors at every iteration. In this way, the path of $\mathbf{d}_t^a$'s will converge to the cycle obeyed by the $\mathbf{d}_t$'s.

Consider an arbitrary vector $\mathbf{d}_1^a$, which we assume to be close enough to the manifold so that the distance between vector $\mathbf{d}_1^a$ and vector $\mathbf{d}_1$ (defined below) is $O(\varepsilon)$. Define

$$k_1^a := \sum_{i=1}^{q_1} d_{1,i}^a, \quad k_2^a := \sum_{i=q_1+1}^{q_1+q_2} d_{1,i}^a, \quad k_3^a := \sum_{i=q_1+q_2+1}^{q_1+q_2+q_3} d_{1,i}^a, \quad \text{and} \quad k_4^a := \sum_{i=q_1+q_2+q_3+1}^{m} d_{1,i}^a.$$

Assume that $(k_1^a, k_2^a, k_3^a)^T$ is $O(\varepsilon)$ close to either $\mathbf{d}_1^{cyc}, \mathbf{d}_2^{cyc}, \mathbf{d}_3^{cyc}, \mathbf{d}_1^{cyc'}, \mathbf{d}_2^{cyc'}$, or $\mathbf{d}_3^{cyc'}$ from Section 4. Since we are permitted to freely shuffle the rows and columns of $\mathbf{M}$ without loss of generality, we assume that $(k_1^a, k_2^a, k_3^a)^T$ is $O(\varepsilon)$ close to $\mathbf{d}_1^{cyc}$, i.e., that k_1^a is $O(\varepsilon)$ close to $k_1 := (3-\sqrt{5})/4$, k_2^a is close to $k_2 := (\sqrt{5}-1)/4$, k_3^a is close to $k_3 := 1/2$, and k_4^a is $O(\varepsilon)$. Now we will carefully choose a corresponding vector $\mathbf{d}_1$ on the manifold, namely

$$d_{1,i} := \left\{ \begin{array}{ccc} \frac{k_1}{k_1^a} d_{1,i}^a & \text{if} & i \le q_1 \\ \frac{k_2}{k_2^a} d_{1,i}^a & \text{if} & q_1 < i \le q_1+q_2 \\ \frac{k_3}{k_3^a} d_{1,i}^a & \text{if} & q_1+q_2 < i \le q_1+q_2+q_3 \\ 0 & \text{otherwise} & \end{array} \right\}. \tag{13}$$

Define $\mathbf{a}_1$ as follows:

$$\varepsilon a_{1,i} := d_{1,i}^a - d_{1,i} = \left\{ \begin{array}{ccc} \left(\frac{k_1^a}{k_1} - 1\right) d_{1,i} & \text{if} & i \le q_1 \\ \left(\frac{k_2^a}{k_2} - 1\right) d_{1,i} & \text{if} & q_1 < i \le q_1+q_2 \\ \left(\frac{k_3^a}{k_3} - 1\right) d_{1,i} & \text{if} & q_1+q_2 < i \le q_1+q_2+q_3 \\ d_{1,i}^a & \text{otherwise} & \end{array} \right\}.$$

The assumption that $\mathbf{d}_1^a$ is sufficiently close to the manifold amounts to the assumption that $\|\mathbf{a}_1\|_1$ is $O(1)$. It is important to note that each of the four pieces of $\mathbf{a}_1$ is proportional to a piece of $\mathbf{d}_1$, and thus proportional to a piece of $\mathbf{d}_1^a$. Recalling that $r_1 = r_2 = r_3 = r = (\sqrt{5}-1)/2$, we have:

$$\begin{aligned} r_1^a &= (\mathbf{d}_1^a \mathbf{M})_1 = (\mathbf{d}_1 \mathbf{M})_1 + \varepsilon(\mathbf{a}_1^T \mathbf{M})_1 = r + \varepsilon(\mathbf{a}_1^T \mathbf{M})_1, \quad \text{so} \\ \frac{1}{1-r_1^a} &= \frac{1}{1-r-\varepsilon(\mathbf{a}_1^T \mathbf{M})_1} = \frac{1}{(1-r)\left(1-\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1-r}\right)} \\ &= \frac{1}{1-r}\left(1+\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1-r}\right) + O(\varepsilon^2) \\ \frac{1}{1+r_1^a} &= \frac{1}{1+r+\varepsilon(\mathbf{a}_1^T \mathbf{M})_1} = \frac{1}{(1+r)\left(1+\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1+r}\right)} \\ &= \frac{1}{1+r}\left(1-\frac{\varepsilon(\mathbf{a}_1^T \mathbf{M})_1}{1+r}\right) + O(\varepsilon^2). \end{aligned}$$

According to the iterated map, for $i \le q_1$, removing terms of $O(\varepsilon^2)$,

$$\begin{aligned} d^a_{2,i} &= \frac{d^a_{1,i}}{1-r^a_1} \\ &\approx \frac{d_{1,i}}{1-r}\left(1+\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}\right)+\frac{\varepsilon a_{1,i}}{1-r} \\ &= \frac{d_{1,i}}{1-r}\left[1+\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{\varepsilon a_{1,i}}{d_{1,i}}\right] \\ &= \frac{d_{1,i}}{1-r}\left[1+\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\left(\frac{k^a_1}{k_1}-1\right)\right] \\ &= d_{2,i}\left[\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{k^a_1}{k_1}\right]. \end{aligned} \tag{14}$$

Since the term in brackets does not depend on i, we know $d^a_{2,i}$ is proportional to $d_{2,i}$ for $i \le q_1$. Recall that for the 3-cycle, the edge of weak classifier 1 must be 0 at iteration 2. Thus, $(\mathbf{d}^a_2\mathbf{M})_1 = 0$, and $\sum_{i=1}^m d^a_{2,i} = 1$, and as before:

$$-\sum_{i=1}^{q_1} d^a_{2,i} + \sum_{i=q_1+1}^{m} d^a_{2,i} = 0 \qquad \text{and} \qquad \sum_{i=1}^{q_1} d^a_{2,i} + \sum_{i=q_1+1}^{m} d^a_{2,i} = 1, \text{ so}$$

$$\sum_{i=1}^{q_1} d^a_{2,i} = \frac{1}{2} \quad \text{and we also have} \quad \sum_{i=1}^{q_1} d_{2,i} = \frac{1}{2}. \tag{15}$$

Combining (14) and (15),

$$\frac{1}{2} = \sum_{i=1}^{q_1} d^a_{2,i} \approx \left[\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{k^a_1}{k_1}\right]\sum_{i=1}^{q_1} d_{2,i} = \left[\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1-r}+\frac{k^a_1}{k_1}\right]\frac{1}{2},$$

thus

$$d^a_{2,i} \approx d_{2,i} \quad \text{for} \quad i \le q_1. \tag{16}$$

A similar calculation for $q_1 < i \le q_1 + q_2$ yields

$$\begin{aligned} d^a_{2,i} &= \frac{d^a_{1,i}}{1+r^a_1} \\ &\approx \frac{d_{1,i}}{1+r}\left(1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}\right)+\frac{\varepsilon a_{1,i}}{1+r} \qquad (17) \\ &= \frac{d_{1,i}}{1+r}\left[1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\frac{\varepsilon a_{1,i}}{d_{1,i}}\right] \\ &= \frac{d_{1,i}}{1+r}\left[1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\left(\frac{k^a_2}{k_2}-1\right)\right] \\ &= d_{2,i}\left[-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\frac{k^a_2}{k_2}\right]. \qquad (18) \end{aligned}$$

And similarly, for $q_1 + q_2 < i \le q_1 + q_2 + q_3$,

$$d^a_{2,i} \approx \frac{d_{1,i}}{1+r}\left(1-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}\right)+\frac{\varepsilon a_{1,i}}{1+r} = d_{2,i}\left[-\frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}+\frac{k^a_3}{k_3}\right]. \tag{19}$$

For the remaining $q_1+q_2+q_3 < i \le m$,

$$d^a_{2,i} = \frac{\varepsilon a_{1,i}}{1+r^a} \approx \frac{\varepsilon a_{1,i}}{1+r}\left[1 - \frac{\varepsilon(\mathbf{a}_1^T\mathbf{M})_1}{1+r}\right] \approx \frac{\varepsilon a_{1,i}}{1+r}. \tag{20}$$

This last set of components will always shrink as t increases, since the last q_4 examples are always correctly classified. From (16), (18), (19), and (20), we can see that each of the first three sections of $\mathbf{d}_2^a$ is proportional to the corresponding section of $\mathbf{d}_2$ to $O(\varepsilon^2)$, and that the last section of $\mathbf{d}_2^a$ is vanishing.

Now calculating the vector $\mathbf{a}_2$ to $O(\varepsilon)$, incorporating equations (16), (17), (19), and (20):

$$\varepsilon a_{2,i} = d^a_{2,i} - d_{2,i} \approx \left\{ \begin{array}{ccc} 0 & \text{if} & i \le q_1 \\ -\frac{\varepsilon d_{1,i}(\mathbf{a}_1^T\mathbf{M})_1}{(1+r)^2} + \frac{\varepsilon a_{1,i}}{1+r} & \text{if} & q_1 < i \le q_1+q_2+q_3 \\ \frac{\varepsilon a_{1,i}}{1+r} & \text{otherwise} & \end{array} \right\}.$$

We will now show that $\mathbf{a}_1$ undergoes a contraction via the iterated map.

$$\begin{aligned} \varepsilon\|\mathbf{a}_2\|_1 = \|\mathbf{d}_2^a - \mathbf{d}_2\|_1 &\le \varepsilon\left[\frac{\left(\sum_{i=q_1+1}^m d_{1,i}\right)|(\mathbf{a}_1^T\mathbf{M})_1|}{(1+r)^2} + \frac{\|\mathbf{a}\|_1}{1+r}\right] + O(\varepsilon^2) \\ &\le \varepsilon\left[\frac{\left(\sum_{i=q_1+1}^m d_{1,i}\right)\|\mathbf{a}_1\|_1}{(1+r)^2} + \frac{\|\mathbf{a}_1\|_1}{1+r}\right] + O(\varepsilon^2). \end{aligned}$$

Aside, we note $\left(\sum_{i=q_1+1}^m d_{1,i}\right) = 1/2 + (\sqrt{5}-1)/4 = (\sqrt{5}+1)/4$. Also, $r = (\sqrt{5}-1)/2$, so $1/(1+r) = (\sqrt{5}-1)/2$. Now,

$$\begin{aligned} \|\mathbf{a}_2\|_1 &\le \|\mathbf{a}_1\|_1\left[\frac{1+\sqrt{5}}{4}\frac{(\sqrt{5}-1)^2}{4} + \frac{\sqrt{5}-1}{2}\right] + O(\varepsilon) \\ &= \|\mathbf{a}_1\|_1\frac{3(\sqrt{5}-1)}{4} + O(\varepsilon) < \|\mathbf{a}_1\|_1. \end{aligned}$$

Thus, we have shown a contraction at the first iteration. The same calculation (with indices changed accordingly) is valid for each iteration, since the relation between $\mathbf{d}_2^a$ and $\mathbf{d}_2$ is now the same as the relation between $\mathbf{d}_1^a$ and $\mathbf{d}_1$, that is, each section of $\mathbf{d}_2^a$ is proportional to the corresponding section of $\mathbf{d}_2$ from (16), (18), and (19) (and the last section vanishes). Since the calculation is valid for each iteration, there is a contraction to the manifold at every iteration. Hence, the manifold is stable.

Appendix C. 4 Cycle for the 3×3 matrix in the Non-Optimal Case

In this appendix, we prove the following theorem:

Theorem 8 *For the 3×3 matrix analyzed in Section 4, AdaBoost in the non-optimal case may produce a 4-cycle which yields a maximum margin solution.*

Proof We will show the existence of such a 4-cycle by presenting its coordinates, and omit a proof of stability. One coordinate on the cycle is given by

$$\mathbf{d}_1^{cyc} = \left(\frac{1}{2}, \frac{1}{2} - \frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4}\right)^T.$$

The only choice for j_1 is $j_1 = 2$, and the edge value is $(\mathbf{d}_1^{cycT}\mathbf{M})_2 = 1/\sqrt{2}$, which is larger than 1/3. Now, we compute $\mathbf{d}_2^{cyc}$ using the iterated map:

$$\mathbf{d}_2^{cyc} = \left(\frac{1}{2+\sqrt{2}}, \frac{1}{2}, \frac{1}{\sqrt{2}} - \frac{1}{2}\right)^T.$$

We now choose $j_2 = 1$ even though it is not the optimal choice. We are justified in this choice, since the edge is $(\mathbf{d}_2^{cycT}\mathbf{M})_1 = \sqrt{2} - 1 > 1/3$. Now iterating again, we obtain $\mathbf{d}_3^{cyc}$:

$$\mathbf{d}_3^{cyc} = \left(\frac{1}{2}, \frac{\sqrt{2}}{4}, \frac{1}{2} - \frac{\sqrt{2}}{4}\right)^T.$$

Again, we must choose $j_3 = 3$ since it is the only permissible edge, $(\mathbf{d}_3^{cycT}\mathbf{M})_3 = 1/\sqrt{2}$. Iterating,

$$\mathbf{d}_4^{cyc} = \left(\frac{1}{2+\sqrt{2}}, \frac{1}{\sqrt{2}} - \frac{1}{2}, \frac{1}{2}\right)^T.$$

With the choice $j_4 = 1$, the edge value is $(\mathbf{d}_4^{cycT}\mathbf{M})_1 = \sqrt{2} - 1 > 1/3$, and the next iterate of the map will yield $\mathbf{d}_1^{cyc}$. We have completed the proof. ■

References

Leo Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

Bruno Caprile, Cesare Furlanello, and Stefano Merler. Highlighting hard patterns via adaboost weights evolution. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems, Lecture Notes in Computer Science 2364*, pages 72–80. Springer, 2002.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1/2/3):225–254, 2002.

N. Duffy and D. Helmbold. A geometric approach to leveraging weak learners. In *Computational Learning Theory: Fourth European Conference, EuroCOLT '99*. Springer-Verlag, 1999.

Yoav Freund and Robert 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, August 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.

Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1), February 2002.

T. Y. Li and J. A. Yorke. Period 3 implies chaos. *Amer. Math. Monthly*, 82(10):985–992, 1975.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, 2000.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, LNCS, pages 119–184. Springer Verlag, 2003.

J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3): 287–320, 2001.

Gunnar Rätsch and Manfred Warmuth. Efficient margin maximizing with boosting. unpublished manuscript, 2002.

Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, August 2004.

Cynthia Rudin. *Boosting, Margins and Dynamics*. PhD thesis, Princeton University, April 2004.

Cynthia Rudin, Ingrid Daubechies, and Robert E. Schapire. On the dynamics of boosting. In *Advances in Neural Information Processing Systems 16*, 2004a.

Cynthia Rudin, Robert E. Schapire, and Ingrid Daubechies. Analysis of boosting algorithms using the smooth margin function : A study of three algorithms. Submitted, 2004b.

Cynthia Rudin, Robert E. Schapire, and Ingrid Daubechies. Boosting based on a smooth margin. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, pages 502–517, 2004c.

Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

Abraham Wyner. Boosting and the exponential loss. In *Proceedings of the Ninth Annual Conference on AI and Statistics*, 2002.